Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____О.В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Алгоритмы и программно-аппаратные средства самоадаптивного управления встраиваемыми системами
тема

09.04.01 «Информатика и вычислительная техника»
код и наименование направления

09.04.01.06 «Микропроцессорные системы»
код и наименование магистерской программы

Научный руководитель	_____	проф., зав. каф. ВТ, канд. техн. наук.	О.В. Непомнящий
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Н.А,Латышонок
	подпись, дата		инициалы, фамилия
Рецензент	_____	доц. каф. РиТК канд. техн. наук	Н.Н,Ткачев
	подпись, дата	должность, ученая степень	инициалы, фамилия
Нормоконтролер	_____	доц., канд. техн. наук	В.И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность работы. Современные встраиваемые системы (ВС) управления представляют собой результат междисциплинарного проектирования. Известно, что номенклатура и диапазон применения таких систем довольно широки: автоматизация производств, системы управления и контроля, автодорожные и транспортные системы, космические системы, авионика и ракетостроение, военно-промышленные комплексы и др. [1,2,3]. Традиционное представление вычислительной элементной базы таких систем выходит далеко за границы описания только конструкции и схемотехники, затрагивая все больше вопросы усложнения алгоритмов работы, повышения уровня гибкости и абстракции проектирования программного обеспечения (ПО) [2].

Технология адаптивного управления также является обширной темой, наиболее часто такие технологии применяются во встраиваемых системах обработки изображений, поиска и анализа различных объектов на основе получения визуальной информации [4]. Данные системы представляют собой аппаратно-программные комплексы (АПК) и предназначены для идентификации в режиме реального времени объектов контроля с помощью распознавания образов на изображениях, полученных посредством видеокамер [5].

В настоящее время существует несколько основных подходов к разработке программного обеспечения таких систем. Один из них основывается на использовании библиотек процедур, реализующих различные алгоритмы обработки и обнаружения объектов на изображении [6]. Обычно такие библиотеки содержат несколько сотен процедур которые разделяются на группы: геометрические преобразования изображений, фильтрация, обнаружение объектов по некоторым признакам и т.д. [5]. К библиотекам такого рода относятся AForge.NET [7], VXL [8], LTI и библиотека OpenCV [9],

которую стоит отметить особенно, т.к. она является библиотекой с открытым исходным кодом и содержит значительное количество встроенных алгоритмов обработки изображений. Разработчику предоставляется возможность самостоятельно выбрать интересующие его процедуры и интегрировать в программное обеспечение разрабатываемой встраиваемой системы.

В основе другого подхода лежит идея использования специализированных программно-аппаратных комплексов. Системы, построенные по этому принципу, уже содержат необходимый набор процедур обработки под некую узкоспециализированную задачу, но кроме того позволяют конечному пользователю настраивать некоторые заранее предопределенные параметры функционирования [10].

Исследованиям теории управления означенных систем посвящены работы Ю.М. Баяковского, У. Претта, Т.С. Хуанга, Д. А. Форсайта, Р. Гонсалеса, Д.Ю. Буряка, Значительный вклад в разработку методов и алгоритмов адаптивного управления внесли Д.Л. Фогель. Б.А. Алпатов, С.Ю. Желтов, М.Н. Красильщиков, Г.Г. Себряков, В.Д. Курганов, В.К. Баклицкий.

Тем не менее, известные работы не позволяют однозначно утверждать о создании эффективных, с современной точки зрения, подходов к разработке программного обеспечения для адаптивного управления встраиваемыми системами обработки изображений. Предварительный анализ позволяет выделить ряд принципиальных положений, а именно:

- известно большое число алгоритмов обработки изображений, однако для подавляющего числа этих алгоритмов нельзя заранее предсказать насколько оптимально их применение к некоторой конкретной задаче [11];
- почти для каждой задачи всегда можно подобрать ряд алгоритмов, для ее решения, но выбор наиболее подходящего в большинстве случаев основывается на результатах тестирования. При этом разработчик или пользователь готовой системы вынужден создавать и тестировать

различные комбинации существующих или уже реализованных в системе алгоритмов;

- программно-алгоритмическое обеспечение предъявляет определенные требования к изображениям, содержащим образы детектируемых или анализируемых объектов. Многие встраиваемые системы применяются в сложных нестационарных условиях эксплуатации, например, когда освещенность зоны контроля является переменчивой и непредсказуемой, а объекты контроля обладают различными характеристиками и могут быть загрязнены. При этом, как правило, в системах используются средства формирований изображений (видеокамеры, объективы, средства освещения и т. п.) которые не учитывают особенности работы конкретных алгоритмов распознавания [12]. Это приводит к тому, что в сложных условиях большинство средств формирования изображений представляют низкокачественные изображения объектов, которые малопригодны для работы. Из-за этого эффективность алгоритмов распознавания и, как следствие, самой встраиваемой системы управления значительно снижается;
- адаптивные свойства систем, в основном, нацелены на повышение качества изображений распознаваемых объектов (адаптивное управление параметрами средств формирования изображений), в то время как это свойство не распространяется на сами алгоритмы обработки и не затрагивает более высокий уровень поведения встраиваемой системы [11].

Таким образом, в тематике адаптивного управления встраиваемыми системами, актуальной задачей является разработка новых способов управления встраиваемыми системами. Требуется переход от адаптивного управления аппаратной частью систем, которое по существу представляет собой подстройку той или иной аппаратуры по ограниченным параметрам и для ограниченного набора функций системы к адаптивному управлению системой в

целом. Следовательно, необходима реализация нового принципа программирования таких систем.

На основании вышеизложенного, можно сделать обоснованное предположение о том, что значимые результаты могут быть получены при использовании генетического программирования, например, на базе эволюционных алгоритмов. При этом следует использовать известные решения в сфере обработки изображений, с помощью уже реализованных библиотек алгоритмов [11]. Возникает потребность в интегрированном подходе и проведения исследований для определения технических возможностей адаптивного управления встраиваемыми системами обработки изображений в сочетании с информацией об окружающих условиях, показателями параметров внешней среды, критически влияющих на основные алгоритмы распознавания объектов [12].

То есть, необходимо не только совершенствовать библиотеки или алгоритмы обработки изображений (что несомненно важно), но в первую очередь разрабатывать новые принципы их использования. Кроме того, крайне важно разработать алгоритмы конструирования процедур распознавания с привязкой к конкретным условиям эксплуатации; требуется особое программное решение, позволяющее обеспечить возможность анализа полученных данных, что позволит осуществить разработку адекватных моделей развития встраиваемых систем.

Цель и задачи. Основной целью проводимого исследования является разработка метода генетического программирования для формирования системы обнаружения объекта и программно-аппаратного комплекса на основе модуля адаптивного интеллектуального агента.

Для достижения поставленной цели определены следующие основные задачи исследования:

1) обзор основных алгоритмов детектирования объектов и алгоритмов обработки изображений, реализованных в библиотеке OpenCV, способных

повысить эффективность работы системы распознавания при негативном влиянии окружающей среды на качество изображений;

2) разработка метода автоматического генерирования эффективных процедур фильтрации изображений в виде деревьев решений с использованием адаптации к показателям датчиков окружающей среды и деревьев решений;

3) разработка системы критериев и параметров для оценки качества работы предложенного способа;

4) проведение испытаний полученного подхода на действующем производстве. Сбор, обработка и интерпретация данных о качестве работы программно-аппаратного комплекса.

Научная новизна. Предложен метод генетического программирования, базирующийся на принципе поиска дерева алгоритмов функционирования системы, позволяющий осуществлять выбор процедуры управления ВС в изменяющихся условиях окружающей среды.

Практические результаты работы. Основным, практическим результатом исследования является разработанное алгоритмическое, программное и аппаратное обеспечение промышленного комплекса, примененного на действующем производстве при производстве литых дисков в компании К&К г. Красноярск (справка о внедрении см. приложение В).

Положения, выносимые на защиту

1. Метод генетического программирования алгоритма работы системы распознавания объектов.

2. Метод автоматического генерирования популяции, отличающийся от известных способом адаптивного выбора структуры деревьев решений из процедур библиотеки обработки изображений и показаний датчиков внешней среды, позволяющий повысить эффективность решения задач распознавания объектов в сложных условиях применения.

3. Метод селекции и мутации для генетического программирования, отличающийся от известных селективным способом формирования и отбора

последующих поколений особей в зависимости от приведенной оценки погрешности качества работы системы обнаружения объектов.

4. Алгоритмическое и программное обеспечение комплекса ВС.

5. Результаты практического внедрения аппаратно-программного модуля.

Достоверность обеспечена корректным использованием методов эволюционных алгоритмов и библиотек обработки изображений, непротиворечивостью исследованиям других авторов, использованием сертифицированного программного и аппаратного обеспечения, необходимым объемом экспериментальных исследований, а также подтверждается результатами практических исследований.

Публикации. Основное содержание диссертации отражено в 2 печатных работах (в том числе одна статья в издании, рекомендованном ВАК), получено свидетельство о регистрации программ для ЭВМ №2018612519 от 18 февраля 2018 г.

Личный вклад автора. Основные результаты, изложенные в работе, получены либо непосредственно автором, либо с его участием.

Объем и структура диссертации. Полный объем диссертации составляет 71 страницу, в том числе: 30 иллюстрации, 10 таблиц, 3 приложения, список литературы из 31 источника.

1 Анализ предметной области программного обеспечения встраиваемых систем

1.1 Программное обеспечение встраиваемых систем

Разнообразие задач автоматизации производственных процессов и способов их решения порождает новые требования к управляющим системам. Здесь отдельным классом следует выделить встраиваемые системы. С учетом существующих технических ограничений и выделяемых финансово-временных бюджетов выбор варианта реализации может быть довольно сложным [13, 14].

Как правило, ВС представляют собой программируемые в традиционном стиле обычные компьютеры, отличающиеся, например, от ПК ограниченными вычислительными ресурсами, использованием языков программирования более низкого уровня и ограниченным сервисом инструментальных средств [2]. В подавляющем большинстве случаев такое программирование выполняется в традиционном стиле, не вторгаясь в область аппаратно-зависимых частей и, тем более, не предполагая использование методов адаптации к окружающей среде. Такой подход обладает рядом достоинств, из которых основным можно считать простое и понятное (упрощенное) представление о месте, роли, способах создания ПО. Приемлемое качество проектирования ПО в такой идеологии достигается только в рамках небольших и несложных проектов ВС, где приемлемо использование шаблонных решений. В случае же, когда от программного обеспечения и ВС в целом требуется сложное поведение - такой подход неуместен.

Стремительное развитие цифровой элементной базы, стилей, технологий и парадигм программирования определили необходимость выделения большей части современных ВС в специальную группу, для которой изначально подчеркивается доминирующее значение программных технологий на всех уровнях организации системы. Такие ВС принято называть «Преимущественно

Программными Системами» (англ. «Software Intensive Systems») [15]. Применительно к ВС общего назначения, в которых создание целевых приложений (то есть собственно решение прикладных задач) достаточно четко отделено от создания вычислительной платформы (то есть самой универсальной ВС) данный термин означает в первую очередь то, что акцент в решении задачи смещен в область программной разработки. Помимо усложнения общих принципов и алгоритмов работы получаемого программного обеспечения для ВС, под этим стоит понимать применение новых подходов, призванных повысить интеллектуальные свойства разрабатываемых устройств. На первый план выходят требования к программной части комплексов в их способности самостоятельно принимать решения на основе обработки получаемой информации и адаптивности к окружающим условиям.

Часто такой подход применяют к ВС машинного зрения, типовой задачей которых, например, является обработка и анализ изображения для поиска заданных объектов [16]. Программно-алгоритмическое обеспечение таких систем, в основном состоит из специализированных процедур и библиотек обработки и анализа изображений. Но на современном этапе развития добавляют:

- нейросетевые алгоритмы, предполагающие обучение на примерах искусственной нейронной сети [17];
- алгоритмы машинного зрения на основе моделей нечеткой логики [18];
- методы визуального программирования процедур анализа изображений [12].

Описанные методы, применяющиеся в настоящее время обеспечивают основные требования к тенденциям развития ВС подобного рода, но имеют свои специфические недостатки. Например, нейросетевые алгоритмы требуют составления больших обучающих выборок, причем, при необходимости дообучения такого алгоритма к новым условиям эксплуатации и расширения

обучающей выборки, разработчик будет вынужден вновь провести обучение нейронной сети [17]. Примечательно, что процедуры обработки и анализа обнаружения в искусственных нейронных сетях формируются внутри самой сети и этот процесс в явном виде не доступен человеку, что не предоставляет возможности анализа этого процесса [19].

Во многих случаях полезным подходом является гибридизация, когда в программное обеспечение ВС включается несколько интеллектуальных систем, в том числе и различных типов. Данный подход становится все более распространенным с ростом производительности вычислительной техники. Однако, разработка и настройка даже одной такой системы – нетривиальная задача. Для ее решения необходим эффективный инструментарий и подход к его применению, включающий способность интеллектуального агента правильно применять эти инструменты в зависимости от информации об окружающих условиях. Таким подходом могут быть алгоритмы эволюционного программирования (ЭП), наиболее известными из которых являются генетический алгоритм (ГА) и алгоритм генетического программирования (ГП) [20]. Основоположником первых методов самоадаптации в ЭП были предложены Д. Л. Фогелем. Стоит отметить, что при самоадаптации в ЭП начала генерируется потомок, а затем изменяются значения параметров алгоритма, порядок выполнения этих действий существенное значение [21].

Генетический алгоритм является эффективной процедурой решения сложных задач оптимизации. Он производит поиск решений в гиперплоскости, определяемой бинарной строкой, которой кодируется решение. Задача ГА – определить, какая (какие) точка исследуемой плоскости доставляют экстремум целевой функции. ГП отличается тем, что генотип индивида представлен нелинейно (в виде бинарной строки или массива чисел), а в виде иерархической структуры. Потенциально, разработчик такой программной системы не ограничивает пространство поиска, а только определяет элементы и связи, при помощи которых строятся решения. Элементы, объединяясь в иерархические

структуры, порождают системы с новыми свойствами, поэтому ГП является одним из наиболее привлекательных инструментов поиска и адаптации в пространстве сложных структур [22].

Однако эффективное применение ГА и ГП на данный момент требует глубокого знания теории эволюционного поиска, что ограничивает операции, которыми оперируют данные алгоритмы. Как правило генерируемые программы состоят из простейших операций, которые образуют вместе алгоритм вычислений [23]. Использование более сложных операций, таких как полноценные библиотечные функции и позволило бы существенно расширить возможность применения ГП в различных встраиваемых системах. Одним из решений этой проблемы является разработка самонастраивающихся алгоритмов, которые самостоятельно адаптируются под решаемую задачу и внешние условия, выбирая эффективные функции и их параметры.

1.2 Выводы по главе

Был проведен анализ предметной области применения встраиваемых систем, обозначен актуальный вектор развития их программного обеспечения в плане интеллектуального управления и свойств адаптивности. Отмечено, что, наиболее ярким представителем подобных задач являются интеллектуальной системы обработки изображений. Учитывая массовость применения подобных систем в различных разделах науки и техники, разрабатываемый в рамках ВКР метод предполагается отрабатывать именно на таких системах.

Согласно заданию на ВКР, следует рассмотреть ряд алгоритмов, характерных для данного класса задач. Требуется разработка нового подхода к программированию ВС, обеспечивающего адаптивность и робастность. В основу предлагается заложить принцип генетического программирования и использовать при этом уже имеющиеся библиотеки.

2 Обзор алгоритмов обработки изображений и детектирования объектов

Так как в рамках данной работы рассматривается задача создания метода генерации оптимальной процедуры обработки изображений с помощью эволюционных методов, необходимо выяснить назначение основных групп и типов алгоритмов, составляющих это множество. В настоящей главе выполнен обзор алгоритмов обработки изображений и детектирования объектов библиотеки OpenCV, взятой как элементная основа в разработке программной части модели модуля адаптивного интеллектуального агента.

2.1 Основные виды алгоритмов распознавания объектов

Практическую реализацию в составе библиотеки находят три основные группы алгоритмов распознавания объектов:

- алгоритмы согласованной фильтрации с проверкой совпадения, использующие шаблоны изображений объектов, которые необходимо обнаружить [11];
- алгоритмы статистического распознавания образов, предусматривающие выборку изображений образов объектов с учётом их конкретных характерных признаков [11];
- алгоритмы на основе моделей, с помощью которых сравниваются характерные признаки наблюдаемой цели с хранящимися в памяти встраиваемой системы моделями данного изображения;

Критерием возможности применения данных функций библиотеки в разрабатываемом подходе, является возможность оценки точности их работы [12]. Это необходимо для целевой функции метода генерации оптимальной процедуры обработки изображения, следовательно, в данной работе логично рассматривать только удовлетворяющие критерию алгоритмы.

2.1.1 Алгоритм сопоставления шаблонов

Для распознавания областей на исходном посредством согласованной фильтрации с проверкой совпадения по шаблону, в библиотеке OpenCV реализована функция cv2.MatchTemplate(). Функция накладывает шаблон изображения на текущее изображение и согласно выбранному алгоритму выполняет поиск корреляции между ними [26].

Ниже приведен пример использования функции распознавания простого объекта квадратной формы (рисунок 2.1в)

а) изображение шаблона объекта; б) анализируемое изображение; в) результат поиска обозначен квадратной рамкой на анализируемом изображении

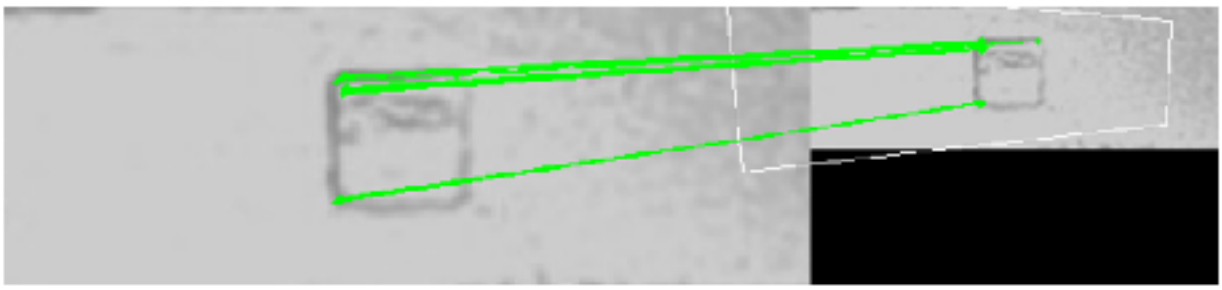
Рисунок 2.1 – Результат применения функции

2.1.2 Метод поиска ключевых точек

Концепция поиска ключевых точек или меток относится к методам, которые нацелены на вычисление абстракций изображения и выделения на нем ключевых особенностей. Данные особенности затем используются для сравнения двух изображений с целью выявления у них общих составляющих. Не существует строго определения того, что такое ключевая особенность изображения. Ею могут быть как изолированные точки, так и кривые или некоторые связанные области. Примерами таких особенностей могут служить грани объектов и углы [24].

Метод позволяет находить точки, строить их описание (дескрипторы) и проводить поиск похожих точек на втором изображении [25]. После чего отфильтровывает неправильно сопоставленные точки, исходя из гипотезы что объект жесткий (недеформируемый) и поэтому все точки на нем должны преобразовываться от одного кадра к другому согласованно. В случае плоского объекта это должно быть перспективное преобразование (гомография) [25].

После того как анализируемые изображения загружены и переданы функции выделения точек, производится поиск матрицы наилучшего перспективного преобразования, осуществляющую это соответствие. На рисунке 2.2 отмечены "правильные" пары особых точек, т. е. пары, подчиняющиеся найденному перспективному преобразованию.



а) выделенные уникальные точки исследуемого изображения сопоставлены с точками шаблона; б) изображения шаблона объекта

Рисунок 2.2 – Результат применения функции detectAndCompue

После того как матрица гомографии найдена, само расположение объекта в пространстве может быть выделено функцией с помощью среднего значения положения всех уникальных точек [27].

2.2 Влияние внешних факторов на работу алгоритмов распознавания

Использование вышеуказанных методов библиотеки OpenCV приводит к появлению специфических погрешностей, которые своим возникновением обязаны влиянием внешних факторов или проблем с аппаратурой системы. Экспериментально доказано, что изменение освещённости в зависимости от времени суток, состояния атмосферы, расстояния до анализируемого объекта – все эти факторы создают помехи, приводящие к ложным тревогам и ошибочным результатам распознавания с недопустимым уровнем ошибки [28].

Рассмотрим более подробно, какие существуют факторы, влияющие на адекватность определения и распознавания объектов на изображении.

1) Изменение уровня освещенности, например, вспышка света или блики от искусственного освещения производственного цеха, приводят к появлению тени, высокой контрастности и экспозиции с фоновой засветкой. Тень искажает форму объекта и создает иллюзию изменчивости геометрических свойств объекта. Освещение в значительной мере влияет на возможность идентифицировать и распознать заданный объект. Для получения наилучшего результата необходимо отслеживать в течение дня изменения интенсивности и направленности солнечного света, корректируя положение камеры и используя дополнительное освещение, во избежание ситуации с фоновой подсветкой. От уровня освещения также зависит такой параметр камеры как значение глубины резкости, оно увеличивается при хорошем освещении. Чем больше показатель глубины резкости, тем больше область, в которой объекты находятся в фокусе (рисунок 2.3).

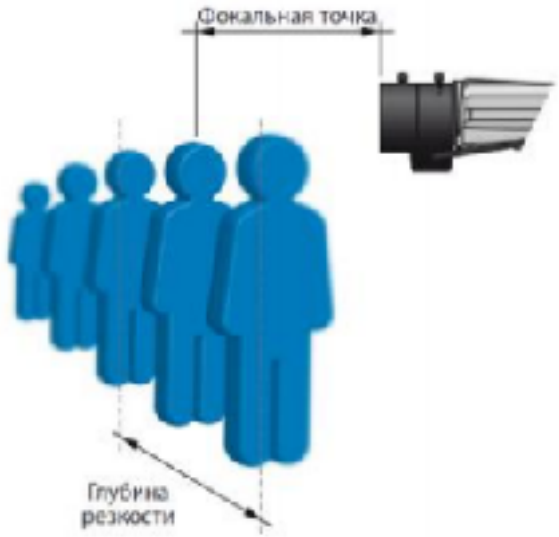


Рисунок 2.3 – Представление глубины резкости

При низком уровне освещенности может появится шум на изображении, что осложняет идентификацию (рисунок 2.4).

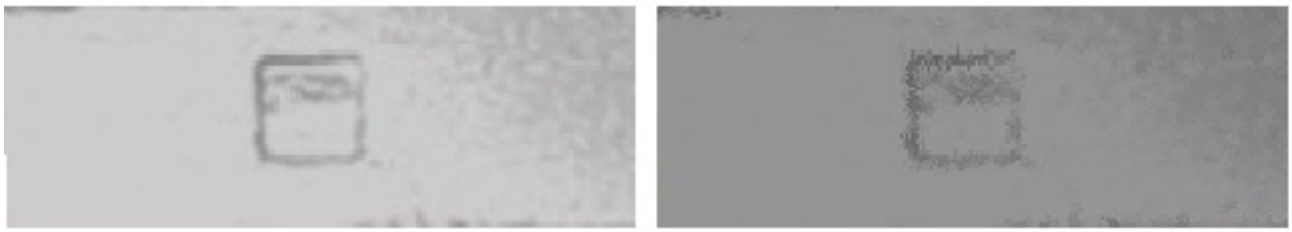


Рисунок 2.4 – Пример шума при низкой освещенности

2) Состояние атмосферы (взвесь частиц, загазованность). Происходит размывание формы объекта. Поверхности предметов, попадающие в кадр, также влияют на идентификацию объектов и могут вызвать ложное срабатывание. Мокрые поверхности, например, усиливают отраженный свет, матовые поглощают некоторую часть отраженного света.

3) Спектральные изменения освещенности, связанные с временем суток. Меняется чувствительность камеры, меняется четкость объектов.

4) Эффекты, связанные с особенностями камеры:

- в следствии продолжительной работы видео оборудования в составе встраиваемой промышленной системы в систему видео объектива могут попадать инородные объекты, такие как частички пыли, смазочно-охлаждающие жидкости и т.д.;
- раскручивание деталей видеокамеры из-за вибрации стоящего промышленного оборудования.

Перечисленные эффекты будут усиливаться со временем, что потребует перенастройки алгоритмов под изменения в качестве получаемых изображений. Для возможности реализации свойств адаптивности разрабатываемого метода, принято решение анализировать вышеуказанные дестабилизирующие факторы внешних условий с помощью ряда датчиков.

2.3 Предобработка изображений средствами библиотеки OpenCV

Несмотря на то, что методы детектирования объектов библиотеки OpenCV сами по себе являются мощными инструментами и для решения тривиальных задач может хватить одной такой функции, это не является достаточным для применения в агрессивных условиях производства. Помимо этих методов новый подход должен бороться с негативными факторами, нейтрализуя эффекты ухудшения качества обрабатываемых изображений посредством, например, фильтров. Фильтры способны изменить получаемые кадры исключая помехи, причем, учитывая особенность подобранных алгоритмов распознавания, делать это следует на все изображения, как шаблонные, так и тестируемые.

2.3.1 Сглаживающие фильтры

Сглаживание или размытие изображения – это одна из самых простых и часто используемых операций обработки изображений. Как правило, размытие

применяется, чтобы уменьшить шум или артефакты, которые обусловлены выбором камеры и внешним освещением. Также сглаживание играет важную роль при необходимости уменьшить разрешение изображения и получить пирамиду изображений разного масштаба (image pyramids) [27].

Разработчики OpenCV реализовали несколько функций размытия изображения. Далее остановимся на некоторых из них, а именно рассмотрим функции GaussianBlur и medianBlur [11]:

Функция GaussianBlur осуществляет размытия с помощью вычисления матрицы свертки изображения с дискретным ядром Гаусса со стандартными отклонениями, равными sigmaX и sigmaY по осям Ox и Oy соответственно. Заметим, что при вызове данной функции накладывается ограничение на параметр kSize. Ширина и высота ядра должны быть положительными и нечетными, либо нулевыми, если размер ядра определяется из стандартных отклонений. Формула ядра Гаусса:

$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}}. \quad (2.1)$$

Далее показан пример результат выполнения функции GaussianBlur (рисунок 2.5).

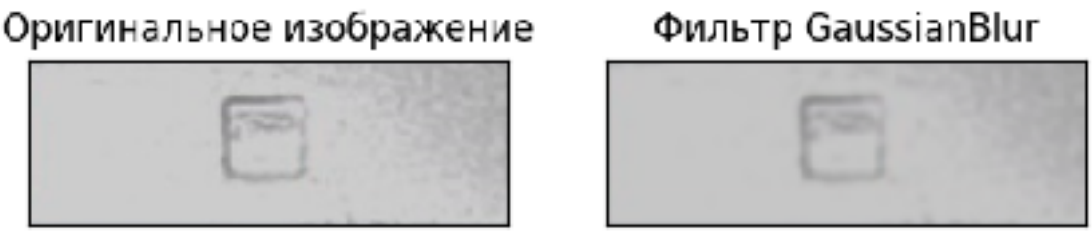


Рисунок 2.5 – Применение функции GaussianBlur

Функция medianBlur обеспечивает размытие посредством применения медианного фильтра. Медианный фильтр обычно используется для уменьшения

шума или «сглаживания» изображения. Выбирается некоторый шаблон, который накладывается на все пиксели изображения. Набор интенсивностей пикселей, которые накрыты шаблоном, сортируются, и выбирается интенсивность, находящаяся в середине отсортированного множества. По сути, определяется медиана в отсортированном наборе данных. Размер шаблона определяется параметром kSize. Фильтр работает с матрицами различного размера, но в отличие от матрицы свёртки, размер матрицы влияет только на количество рассматриваемых пикселей. Ниже представлена работа медианного фильтра для размера ядра равного трём (рисунок 2.6).

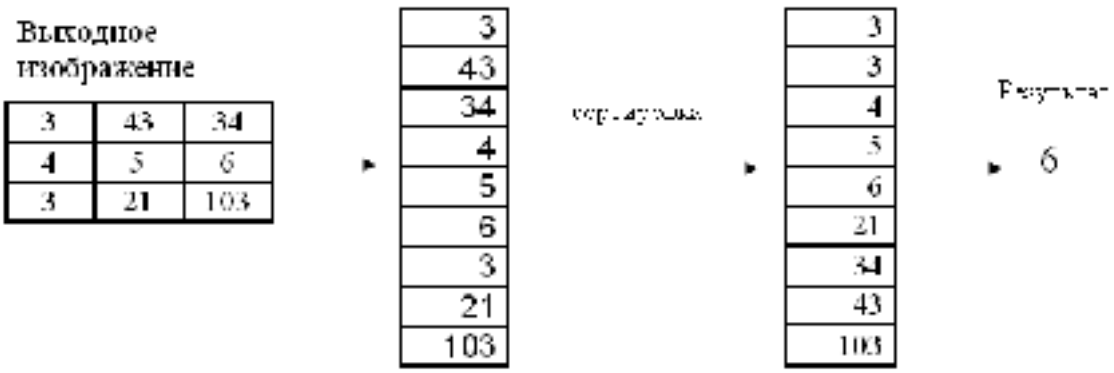


Рисунок 2.6 – Принцип изменения пикселей в работе медианного фильтра

Для текущего пикселя, пиксели, которые «попадают» в матрицу, сортируются, и выбирается средние значение из отсортированного массива. Это значение и является выходным для текущего пикселя. Далее показан пример использования функции medianBlur (рисунок 2.7)

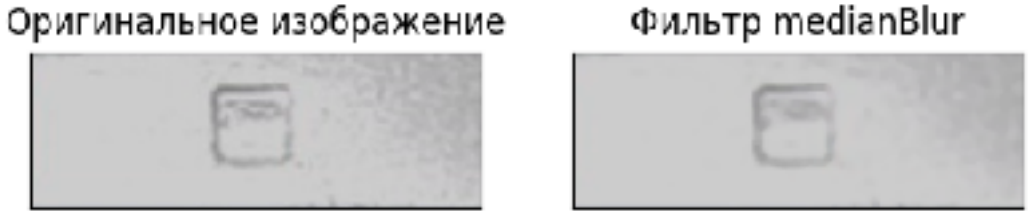


Рисунок 2.7 – Применение функции medianBlur

2.3.2 Фильтры с использованием морфологических преобразований

Дилатация (морфологическое расширение) – свертка изображения или выделенной области изображения с некоторым ядром. Ядро может иметь произвольную форму и размер [1, 2]. При этом в ядре выделяется единственная ведущая позиция (anchor), которая совмещается с текущим пикселем при вычислении свертки. Во многих случаях в качестве ядра выбирается квадрат или круг с ведущей позицией в центре. Ядро можно рассматривать как шаблон или маску. Применение дилатации сводится к проходу шаблоном (рисунок 2.8) по всему изображению и применению оператора поиска локального максимума к интенсивностям пикселей изображения, которые накрываются шаблоном.

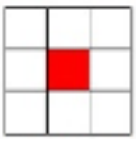
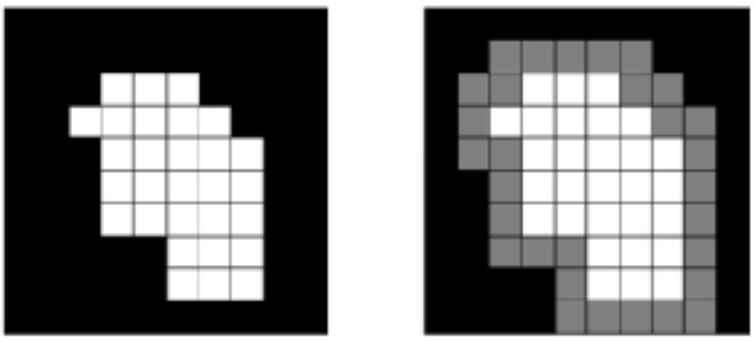


Рисунок 2.8 – Применение морфологических операций

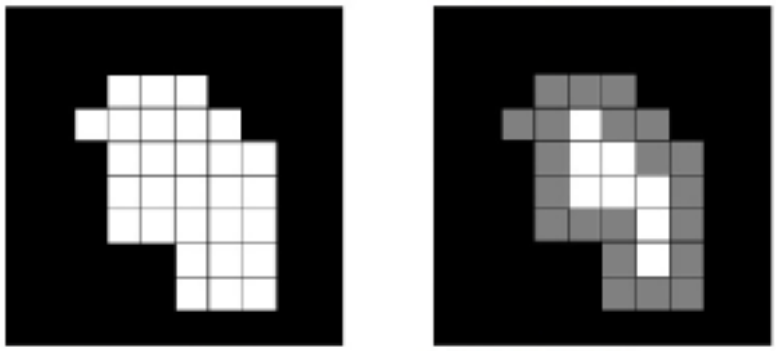
Такая операция вызывает рост светлых областей на изображении (рисунок 2.9). На рисунке серым цветом отмечены пиксели, которые в результате применения дилатации будут белыми.



а) оригинальное изображение; б) результат работы дилатации

Рисунок 2.9 – Применение морфологических операций

Эрозия (морфологическое сужение) – обратная операция. Действие эрозии подобно дилатации, разница лишь в том, что используется оператор поиска локального минимума (рисунок 2.11), серым цветом залиты пиксели, которые станут черными в результате эрозии.



а) оригинальное изображение; б) результат работы эрозии

Рисунок 2.10 – Применение морфологических операций

Результат выполнения программы показан на примере некоторого изображения (рисунки 2.11 и 2.12). Как видно из рисунков, применение эрозии привело к сужению контуров, дилатации – расширению.

Рисунок 2.11 – Результат применения дилатации (слева)

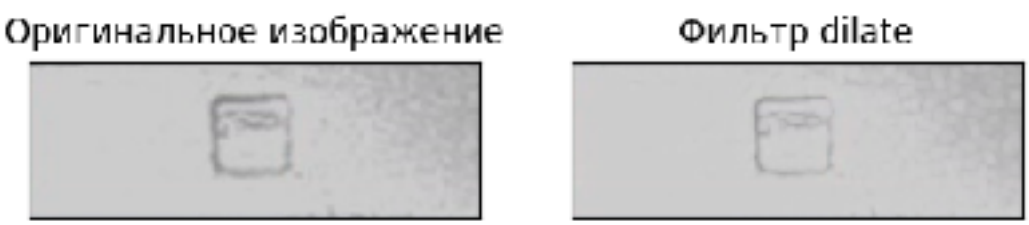


Рисунок 2.12– Результат применения эрозии (слева)

2.3.3 Выравнивание гистограмм

Один из наиболее распространенных дефектов фотографических, сканерных и видео изображений – слабый контраст. Дефект во многом обусловлен ограниченностью диапазона воспроизводимых яркостей. Под контрастом понимается разность максимального и минимального значений яркости. Контрастность изображения можно повысить за счет изменения яркости каждого элемента изображения и увеличения диапазона яркостей.

Выравнивание гистограмм – это один из наиболее распространенных способов. Цель выравнивания состоит в том, чтобы все уровни яркости имели бы одинаковую частоту, а гистограмма соответствовала равномерному закону распределения. В библиотеке OpenCV реализована функция equalizeHist [11], которая обеспечивает повышение контрастности изображения (рисунок 2.13) посредством выравнивания гистограммы [11, 16].

Рисунок 2.13 – Результат выравнивания гистограммы

2.4 Выводы по главе

Приведен обзор алгоритмов детектирования объектов библиотеки OpenCV, были выявлены деструктивные факторы внешней среды, влияющие на качество работы этих алгоритмов. Выбран перечень датчиков, необходимых

для мониторинга указанных факторов. Рассмотрены процедуры обработки изображений, которые необходимо применять совместно с алгоритмами обнаружения в целях нивелирования воздействий внешних условий.

Для повышения эффективности работы алгоритмов распознавания, в зависимости от внешних условий, могут потребоваться различные процедуры фильтрации (сглаживание, морфологическое преобразование и т. п.), а в определённых ситуациях в них нет необходимости. В частных случаях применения алгоритмов распознавания, предварительной обработке изображений следует подвергать как изображение с камеры, так и шаблон распознаваемого объекта. Таким образом, обработку изображения можно представить в виде алгоритма, состоящего из подобранных процедур фильтраций и условий их применения, которыми могут быть показания датчиков внешней среды. В данном случае необходимо использовать такие средства выбора процедур, которые позволили бы полностью автоматизировать процесс генерации оптимальной процедуры. На основании вышеизложенного, можно делать вполне обоснованный вывод о том, что эволюционные алгоритмы, а именно метод генетического алгоритма, позволят получать наибольший эффект в данном случае.

3 Метод генетического программирования для генерации процедур предобработки изображений с помощью деревьев решений

3.1 Эволюционные алгоритмы и деревья решений

В рамках данной работы предлагается новый метод кодирования готовых процедур обработки изображений для генетического программирования, основанный на их представлении с помощью деревьев решений. Это позволит задать ограничения на построение общей процедуры обработки на основе показаний с датчиков окружающей среды. Выбор способа кодирования программы в генетическом алгоритме — один из основных вопросов генетического программирования [29]. Программа должна быть закодирована в таком виде, чтобы в нее можно было вносить случайные изменения (оператор мутации) и объединять два алгоритма в один (оператор скрещивания). Именно это позволит реализовать принцип адаптивности предполагаемого метода. Результатом кодирования будет дерево процедуры.

Согласно предложенному методу, изначально, код программы процедуры обработки изображения преобразуется в дерево разбора, (рисунок 3.1)

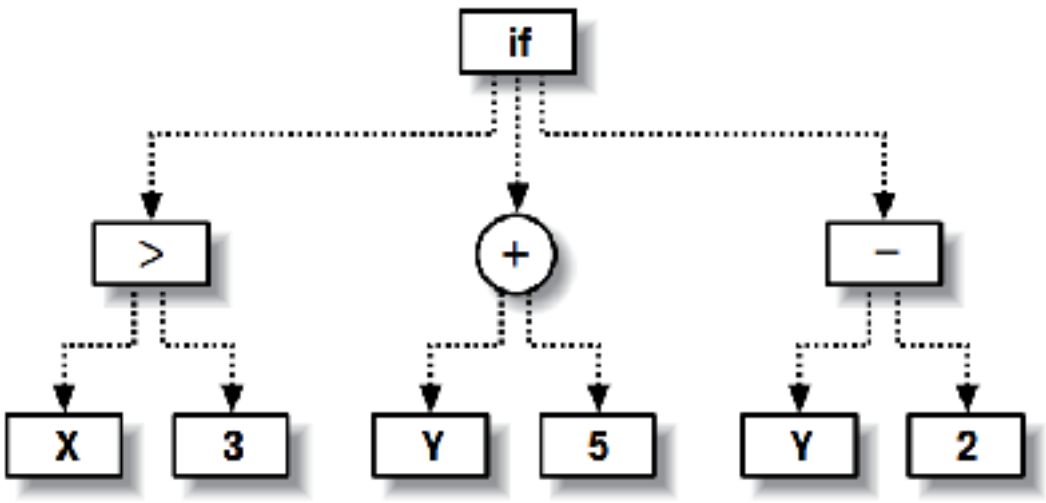


Рисунок 3.1 – Пример программы, представленной в виде дерева

Из рисунка видно, что каждый узел представляет либо операцию над дочерними узлами, либо является «листовым», например, параметром или константой. Так, круглой фигурой представлена операция суммирования двух ветвей, в данном случае значений переменной Y и константы 5. Вычисленная сумма передается вышестоящему узлу, который применяет собственную операцию к своим потомкам. Обратите внимание, что один из узлов соответствует операции if; это означает, что если значение левой ветви равно true, то он возвращает свою центральную ветвь, а если false – то правую ветвь. Дерево может быть рекурсивным, если допускаются ссылки на узлы, расположенные выше. С помощью таких деревьев можно организовывать циклы и более сложные управляющие структуры.

3.2 Представление процедур обработки изображения с помощью деревьев решений

В данном случае, каждое звено дерева имеет один из трёх типов данных:

- логический – ложь или истина;
- числовой – вещественное число от 0 до 1;
- изображение.

Помимо типов данных, каждое звено дерева характеризуется одним из трёх видов:

- функция – звено с потомками;
- переменная (показание одного из датчиков) – лист дерева;
- константа (логическое значение, число или исходное изображение) – лист дерева.

Используемые функциональные звенья приведены в таблице 3.1.

Таблица 3.1 – Функциональные звенья

Функция	**Тип данных**	**Аргументы**	**Результат**
Сравнение («>»)	Логический	1) число 1 2) число 2	Истина, если число 1 больше, чем число 2; в противном случае ложь
Условное выражение («if»)	Изображение	1) условие (логический тип) 2) изображение 1 3) изображение 2	Если условие = истина, то изображение 1; в противном случае изображение 2.
Размытие Гаусса	Изображение	1) **src** – изображение 2) **ksize** – размер ядра (числовой тип)	Изображение **src**, после применения размытия Гаусса с ядром размера ⌊20**ksize**⌋ × ⌊20**ksize**⌋
Медианный фильтр	Изображение	1) **src** – изображение 2) **ksize** – размер ядра (числовой тип)	Изображение **src** после применения размытия медианным фильтром с ядром размера ⌊20**ksize**⌋ × ⌊20**ksize**⌋
Дилатация	Изображение	1) **src** – изображение 2) **ksize** – размер ядра (числовой тип) 3) **ktype** – тип ядра (числовой тип)	Изображение **src** после применения дилатации с круглым ядром размера ⌊20**ksize**⌋ × ⌊20**ksize**⌋. Ядро имеет квадратный тип, если **ktype** < 0,5, и круглый в противном случае.

Окончание таблицы 3.1

Эрозия	Изображение	1) **src** – изображение 2) **ksize** – размер ядра (числовой тип) 3) **ktype** – тип ядра (числовой тип)	Изображение **src** после применения эрозии с круглым ядром размера ⌊20**ksize**⌋ × ⌊20**ksize**⌋. Ядро имеет квадратный тип, если **ktype** < 0,5, и круглый в противном случае.
Выравнивание гистограммы	Изображение	1) **src** – изображение	Изображение **src** после выравнивания гистограммы

3.3 Эволюция процедур обработки изображений

Принцип работы генетического алгоритма заключается в создании набора случайных решений, который называется популяцией. На каждом шаге оптимизации целевая функция вычисляется для всей популяции, в результате чего получается ранжированный список решений. После ранжирования решений, создается новая популяция, которая называется следующим поколением. Сначала в новую популяцию включаются самые лучшие решения из текущей. Этот процесс называется элитизмом [30]. Кроме них, в следующую популяцию входят совершенно новые решения, получающиеся путем модификации наилучших посредством генетических операций, таких как скрещивание и мутация. Конкретные реализации методов представления особей и генетических операций порождают различные разновидности эволюционных алгоритмов. Затем этот процесс повторяется – новая популяция ранжируется и создается очередное поколение. Так продолжается заданное число раз или до тех пор, пока на протяжении нескольких поколений не

наблюдается никаких улучшений. Принцип метода изображен на рисунке 3.2.

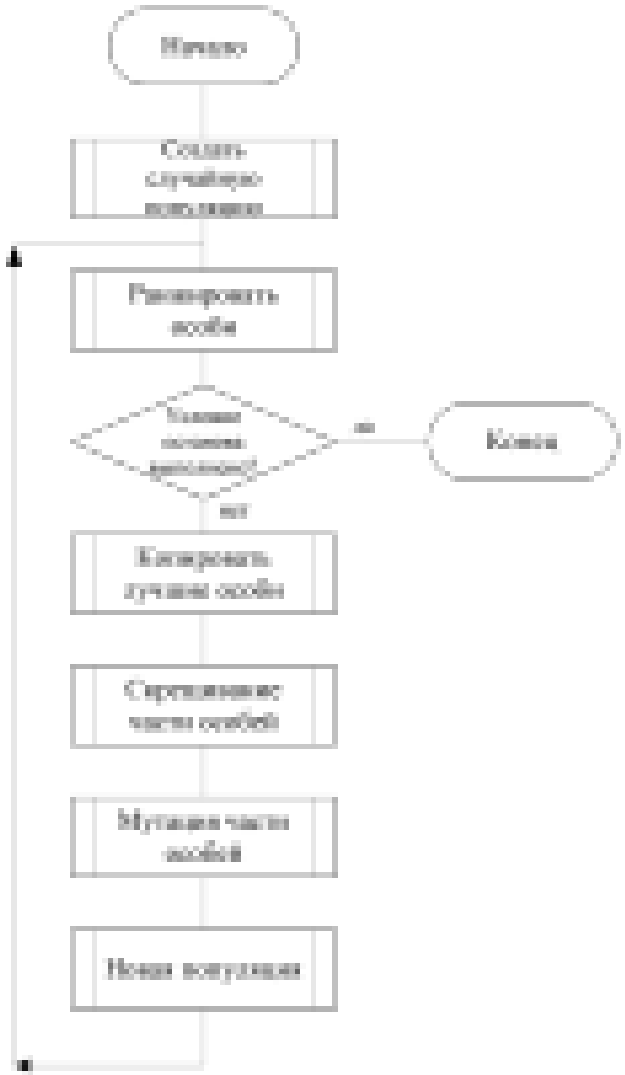


Рисунок 3.2 – Блок-схема генетического алгоритма

Для рассматриваемой задачи генерации системы обнаружения объекта на изображении, в качестве генотипа был выбран вектор $s = \{T_1, T_2, c\}$. Здесь T_1 – дерево решений, соответствующее предобработке изображения с камеры, T_2 – дерево решений, соответствующее предобработке шаблонного изображения, c – параметр алгоритма обнаружения объекта. Полагается, что в результате выполнения генетического алгоритма будет получена система, способная максимально точно устанавливать положение интересующего объекта на изображении или выявлять факт его отсутствия. Схема работы данной системы представлена на Рисунке 3.3.

Рисунок 3.3 – Система обнаружения объекта

Для проверки качества работы системы используется выборка изображений, элементы которой состоят из:

- изображения, полученного с камеры;
- значений датчиков внешней среды в момент получения этого изображения;
- координат центральной точки интересующего объекта на изображении или указания факта его отсутствия.

Система обнаружения применяется к каждому элементу выборки. По результатам применения системы ко всей выборке составляется таблица сопряженности (табл. 3.2).

Таблица 3.2 – Таблица сопряжённости

	Объект присутствует	Объект отсутствует
Объект обнаружен	$N_{ип}$	$N_{лп}$
Объект не обнаружен	$N_{ло}$	$N_{ио}$

Здесь:

$N_{ип}$ – число истинно положительных случаев;

$N_{лп}$ – число ложно положительных случаев;

$N_{ио}$ – число истинно отрицательных случаев;

$N_{ло}$ – число ложно отрицательных случаев.

Кроме того, для истинно положительных случаев также вычисляется погрешность обнаружения объекта ρ_i:

$$\rho_i = \begin{cases} \sqrt{\left(x_i^* - x_i\right)^2 + \left(y_i^* - y_i\right)^2}, \text{если } i \text{ - истинно положительный случай;} \\ 0 \text{ иначе;} \end{cases} \quad (3.1)$$

где:

i – номер элемента выборки;

(x_i^*, y_i^*) – истинное положение центра объекта на i-м изображении (в пикселах);

(x_i, y_i) – найденное положение центра объекта на i-м изображении (в пикселах).

Для идеальной системы обнаружения верно следующее:

- количество ошибок классификации изображений (интересующий объект есть или отсутствует) равно нулю: $N_{лп} = N_{ло} = 0$;
- для каждого истинно положительного случая (с номером i в выборке) погрешность обнаружения равна 0: $\rho_i = 0$.

Для получения системы обнаружения, наиболее близкой к идеальной, необходимо формализовать критерии качества её работы в функции фитнеса.

3.4 Функция фитнеса генетического алгоритма

Функция фитнеса составляется в соответствии со следующими требованиями:

- чем ниже значение функции фитнеса, тем качественнее система обнаружения объекта;
- чем меньше ошибок классификации изображений (ложноположительных и ложноотрицательных случаев), тем ниже значение функции фитнеса;

- чем меньше погрешность обнаружения (расстояние между истинным и найденным положениями интересующего объекта), тем ниже значение функции фитнеса;
- считать результаты обнаружения одинаково плохими при превышении погрешностью ρ_i определённого порога $\rho_{\max}$;
- ложноположительные и ложноотрицательные случаи имеют различные веса.

В соответствии с вышеизложенным, была выбрана следующая функция фитнеса:

$$Q(s) = w_{ип} \sum_{i=1}^{N_{выб}} \rho_{i\,прив} + w_{лп} N_{лп} + w_{ло} N_{ло}, \tag{3.2}$$

где $\rho_{i\,прив}$ – приведённое значение погрешности определения метки в i-м изображении обучающей выборки, $\rho_{i\,прив} \in [0,1]$; вычисляется по формуле (3.2):

$$\rho_{i\,прив} = \left[\frac{\min(\rho_i, \rho_{\max})}{\rho_{\max}} \right]^{\gamma_{ип}}; \tag{3.3}$$

зависимость между ρ_i и $\rho_{i\,прив}$ показана на Рисунке 3.4;

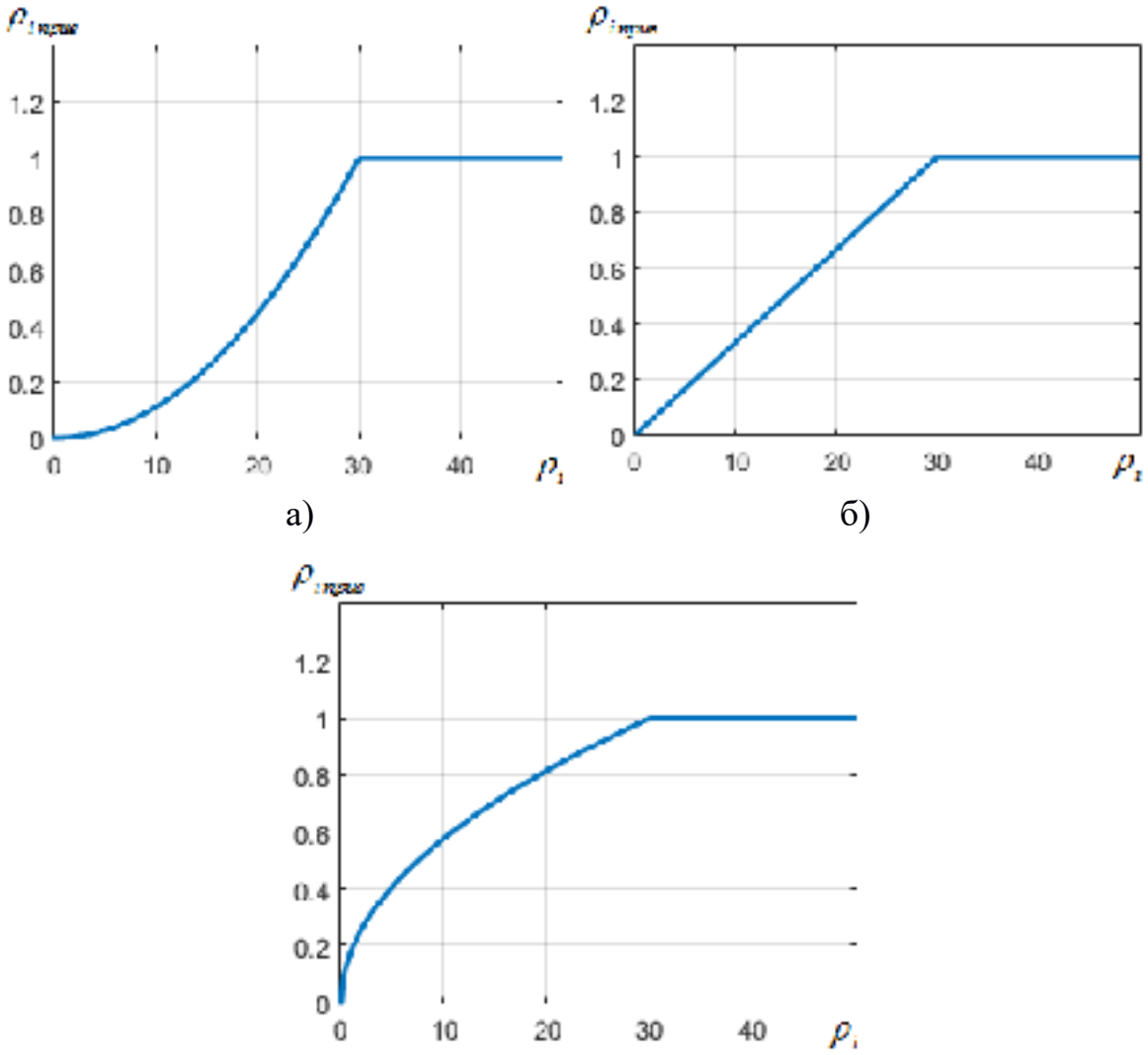
$N_{выб}$ – размер обучающей выборки;

$\gamma_{ип} \in (0, \infty)$ – показатель, регулирующий кривизну зависимости ρ_i и $\rho_{i\,прив}$;

$w_{ип}$ – штрафной коэффициент погрешности в истинно положительных случаях;

$w_{лп}$ – штрафной коэффициент ложно положительных случаев;

$w_{ло}$ – штрафной коэффициент ложно отрицательных случаев.



а) б)

в)

а) $\gamma_{un} > 1$; б) $\gamma_{un} = 1$; в) $0 < \gamma_{un} < 1$

Рисунок 3.4 – Зависимость между $\rho_{i\ прив}$ и ρ_i

3.5 Генерация произвольной процедуры обработки изображения

Для создания начальной популяции генетического алгоритма, а также для оператора мутации необходим алгоритм генерации произвольного дерева решений. В нём случайным образом выбираются звенья и соединяются так, чтобы получилось корректное дерево решений, глубина которого не превышает заданного максимального значения $N_{макс.\ глуб}$. Вероятности выбора каждого звена при заданном типе данных приведены в таблицах 3.3, 3.4 и 3.5.

Таблица 3.3 – Вероятности звеньев логического типа

Звено	**Вероятность**
Сравнение («>»)	0,5
Константа (случайный выбор из 0 и 1)	0,5

Таблица 3.4 – Вероятности звеньев числового типа

Звено	**Вероятность**
Переменная (со случайным номером)	0,5
Константа (случайное число от 0 до 1)	0,5

Таблица 3.5 – Вероятности звеньев с типом данных «изображение»

Звено	**Вероятность**
Условное выражение («if»)	0,2
Фильтр (случайным образом выбирается один из следующих фильтров: размытие Гаусса, медианный фильтр, дилатация, эрозия, выравнивание гистограммы)	0,5
Константа (исходное изображение)	0,3

Блок-схема алгоритма генерации произвольной процедуры предобработки изображения приведена на рисунке 3.3. Она создаёт звено указанного типа данных, и если звено является функциональным, то рекурсивно создаёт ветви для каждого аргумента функции. В качестве типа данных корневого звена нужно указывать изображение.

При генерации начальной популяции данный алгоритм (Рисунок 3.5) запускается дважды для каждой особи: первый раз – для создания дерева T_1, второй – для создания дерева T_2. Кроме этого, случайным образом генерируется параметр c алгоритма обнаружения объекта из диапазона [0, 1].

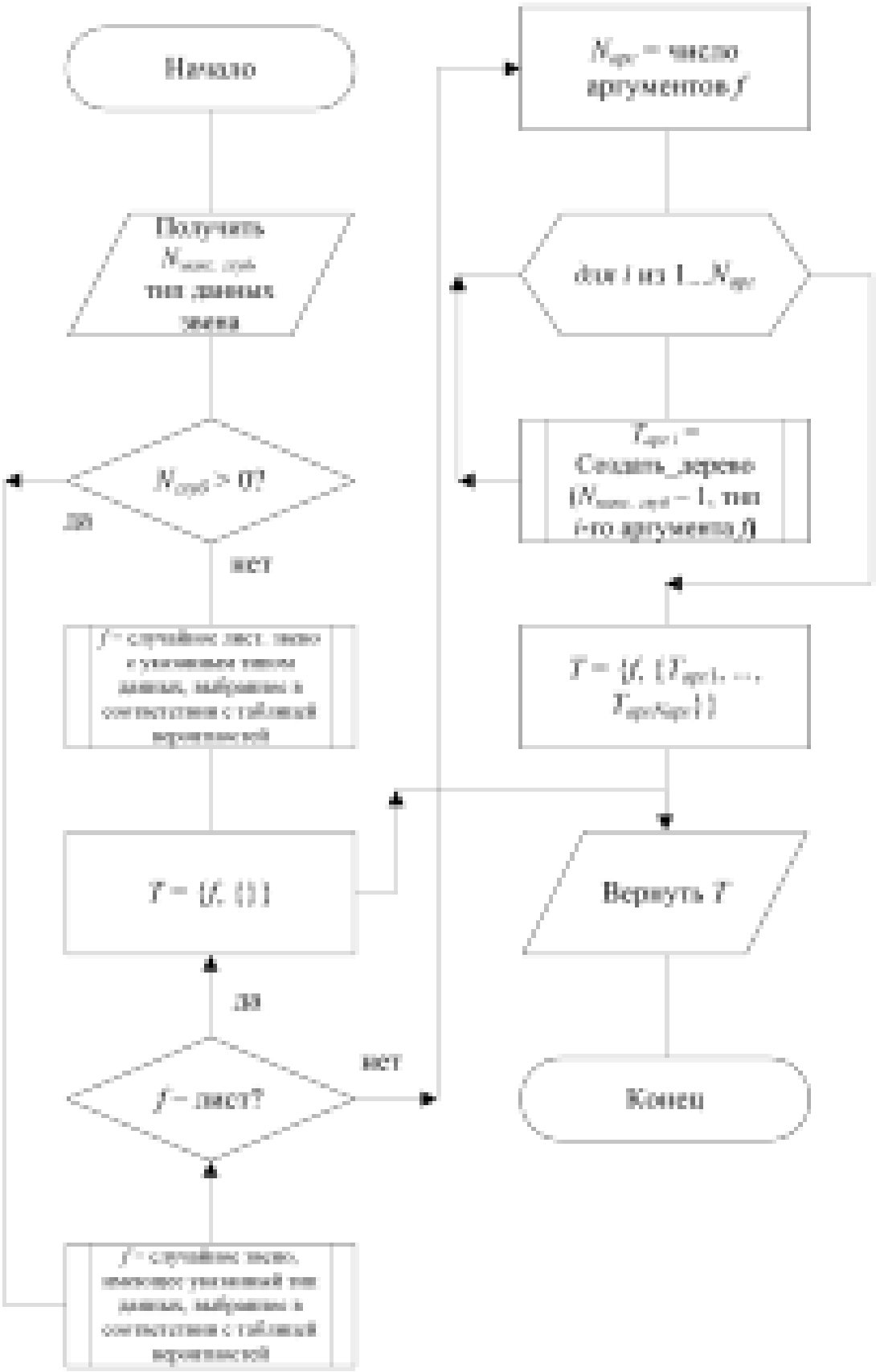


Рисунок 3.5 – Блок-схема алгоритма генерации произвольной процедуры предобработки изображения

3.6 Мутация

Алгоритм мутации использует в своей работе алгоритм генерации произвольной процедуры предобработки изображений, описанный в предыдущем пункте. В дереве решений происходит обход звеньев в глубину, и для каждого звена есть вероятность мутировать, то есть замениться произвольно сгенерированным деревом. В этом случае обход звеньев в этой ветке дерева прекращается. Также имеется ограничение, что глубина мутировавшего дерева не превышает $N_{макс.\ глуб.}$

Блок-схема мутации процедуры предобработки изображения приведена на рисунке 3.4. Этот алгоритм выполняется дважды для мутирующей особи: первый раз – для дерева T_1, второй – для дерева T_2. Кроме того, к параметру c прибавляется случайное число с центрированным нормальным распределением. Если результат окажется меньше 0, то c заменяется значением 0; если результат превысит 1, то c заменяется значением 1.

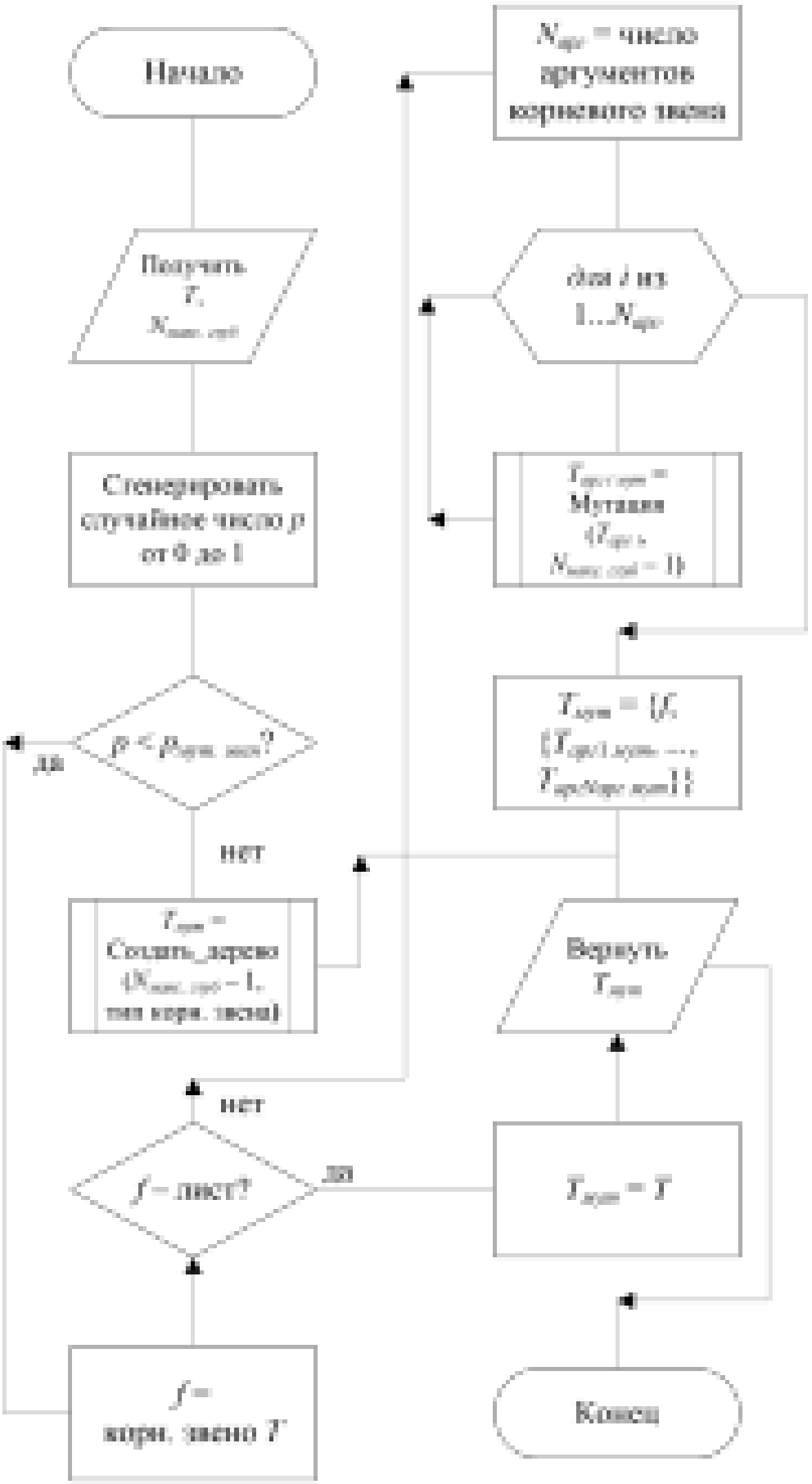


Рисунок 3.6 – Блок-схема алгоритма мутации дерева

3.7 Скрещивание

Скрещивание особей $s_A = \{T_{A1}, T_{A2}, c_A\}$ и $s_B = \{T_{B1}, T_{B2}, c_B\}$ происходит следующим образом: с равной вероятностью извлекаются один за другим соответствующие элементы генотипов особей *A* или *B* и формируется дочерняя особь *C*. Примеры возможных генотипов особи *C*: $s_C = \{T_{B1}, T_{B2}, c_A\}$, $s_C = \{T_{A1}, T_{B2}, c_B\}$, $s_C = \{T_{A1}, T_{A2}, c_A\}$.

3.8 Выводы по главе

Предложен метод генетического программирования алгоритма работы системы распознавания объектов. В основу метода положено представление процедур библиотеки обработки изображений в виде деревьев решений и применение к ним генетических алгоритмов (скрещивания, мутации, отбора).

Разработан и реализован метод автоматического генерирования популяции, отличающийся от известных способом адаптивного выбора структуры деревьев решений из процедур библиотеки обработки изображений и показаний датчиков внешней среды, позволяющий повысить эффективность решения задач распознавания объектов в сложных условиях применения.

Разработан и реализован метод селекции для генетического программирования, отличающийся от известных селективным способом формирования и отбора последующих поколений особей в зависимости от приведенной оценки погрешности качества работы системы обнаружения объектов.

Предложен метод мутации особей для генетического программирования, основанный на разработанном методе автоматического генерирования популяции.

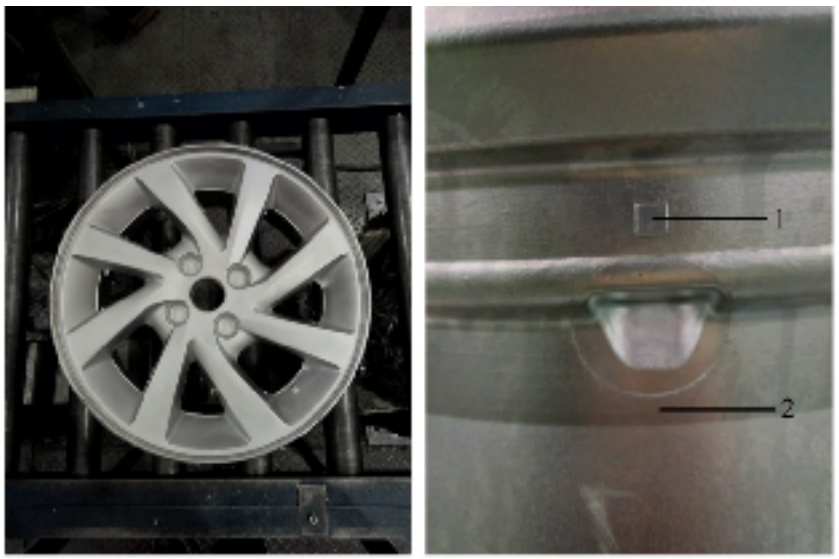
4 Практическое внедрение и результаты экспериментов

Полученные результаты диссертационных исследований были внедрены и испытаны на действующем производстве. Так в цехе по производству литых дисков ООО «K&K», на автоматической линии механической обработки изделий (рисунок 4.1) был развернут стенд комплекса распознавания специальных меток ориентации автомобильного диска.

Рисунок 4.1 – Автоматическая линия производства

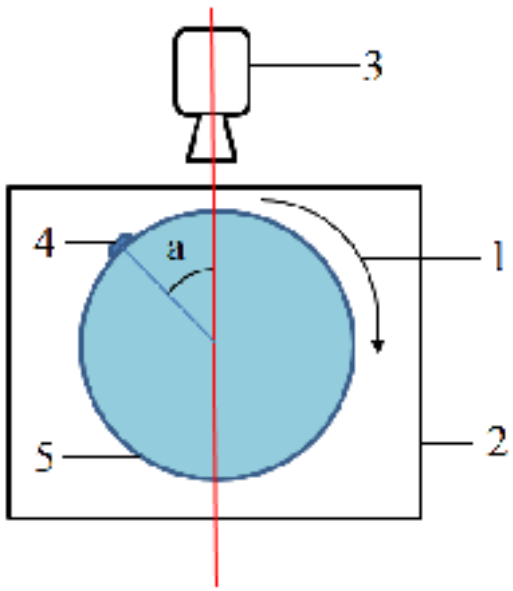
При обработке на автоматической линии изделие последовательно перемещается от одной машины к другой. Оно устанавливается и закрепляется в определенном положении для обработки с помощью промышленного манипулятора, именно для управления этим процессом в линию встроена система технического зрения, отвечающая за правильную ориентацию изделий в захвате промышленного робота. Для контроля положения диска, на его боковой стороне (реборде) отлита специальная физическая метка квадратной формы (рисунок 4.2б). Перед каждой технологической операцией, изделие помещается на специальный стенд (рисунок 4.3), где установлен объектив встраиваемой системы обработки изображений, она должна обнаружить физическую метку и сообщить координаты метки относительно кадра,

полученные координаты будут переданы промышленному контроллеру (ПЛК) в качестве управляющего сигнала для последующих манипуляций.



а) вид сверху; б) вид сбоку: 1) метка ориентации; 2) реборда диска

Рисунок 4.2 – Автомобильный диск



1) направление вращения диска; 2) поворотный стол стенда; 3) объектив IP-камеры; 4) метка ориентации диска; 5) автомобильный диск

Рисунок 4.3 – Стенд анализа ориентации дисков

Учитывая, что помимо довольно непростой типовой задачи распознавания и обнаружения объекта по визуальному изображению, задачу еще более усложняет окружающая среда промышленного производства. Появляется возможность проверки работы адаптивного качества разработанного метода.

4.1 Аппаратная часть стенда

Основной состав системы определяется в соответствии с целью применения разработанных алгоритмов и методов обнаружения заданного объекта. Основной набор датчиков определяется в соответствии с перечнем измеряемых показателей, определённом в главе 2. Регистрация каждого из показателей осуществляется при помощи соответственного датчика, а изображение анализируемого объекта поступает с IP-камеры (Рисунок 4.4).

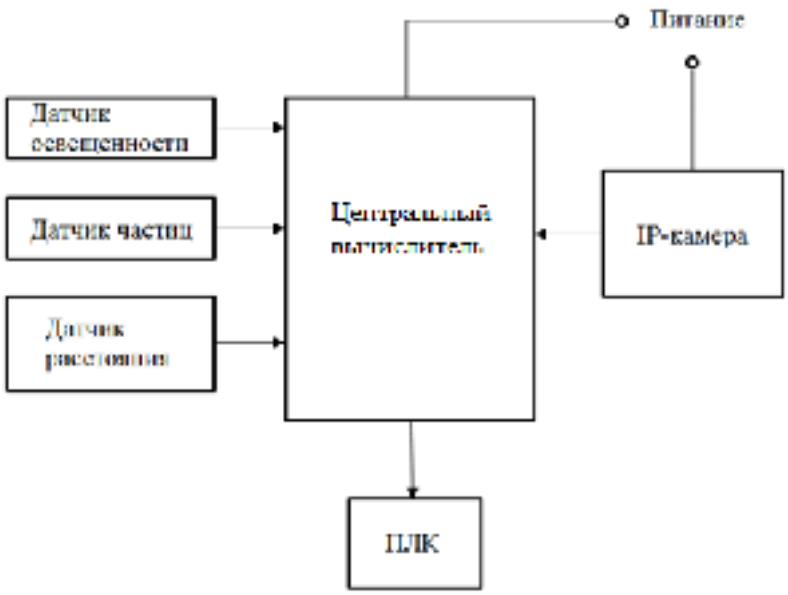


Рисунок 4.4 – Структурная схема стенда

Отметим, что датчик расстояния необходим в составе аппаратуры для измерения расстояния до объекта, поскольку последний находится на боковой стороне автомобильного диска, из-за разных диаметров изделий, может

меняться расстояние от диска до объектива камеры.

4.1.1 Центральный вычислитель

Для снятия данных с датчиков, получения изображения с IP-камеры, последующей обработки изображения и отправки результата в ПЛК по TCP/IP протоколу использован промышленный одноплатный компьютер EPI-945GSE-A1-01R на базе микропроцессора Inetl ATOM N270 и операционной системы Debian Linux. Особенностью данного микрокомпьютера стало наличие контактов GPIO, обеспечивающих возможность подключения датчиков стенда по шине I2C. Внешний вид устройства изображен на рисунке 4.5, а характеристики указаны в таблице 4.1.

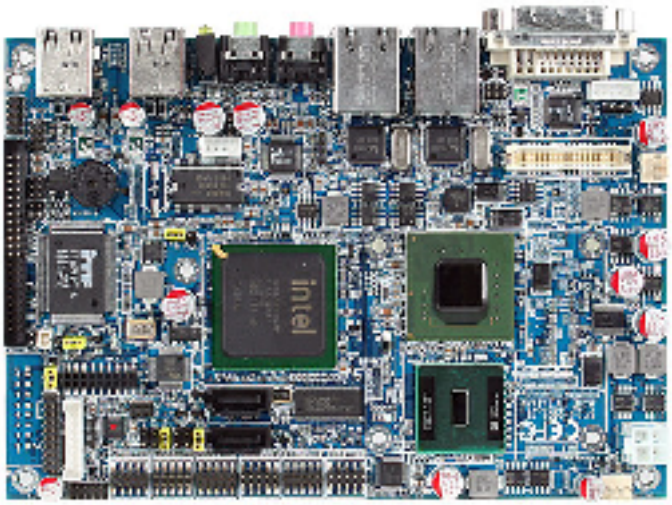


Рисунок 4.5 – Одноплатный компьютер EPI-945GSE-A1-01R

Таблица 4.1 – Основные технические характеристики ЭВМ

CPU	Intel Atom N270 1.6GHz
BIOS	Award 4 Mbit Flash BIOS
GPU	Intel 945GSE
Интерфейсы	1 x EIDE, 2 x SATA, 1 x KB & Mouse (Optional) 5 x RS-232, 1 x RS-232/ 422/ 485 8-bit GPI, 8-bit GPO

Окончание таблицы 4.1

Память (SDRAM, общая)	One 200-pin SODIMM Socket Supports Up to 2GB DDR2 400/ 533 SDRAM
Максимальная рабочая частота	1.6 ГГц
Порты USB 2.0	6
Видеовыход	DVI + LVDS
Сеть	Ethernet-порт RJ45 10/100 Мбит/с
Класс защиты	IP-52
Энергопотребление	700 мА (3,5 Вт)
Питание	5 В через порт micro-USB или GPIO
Размеры	85,6x56x21 мм
Масса	54 г
Температурный режим	от -40 ° С до 85 °

4.1.2 Датчик освещенности

В качестве используемого датчика освещенности был выбран цифровой датчик BH1750. Преимущество датчика среди аналогов его категории точности заключается поддержке работы с шиной I2C и переводе показаний в единицы измерения Люкс. Внешний вид датчика изображен на рисунке 4.6. Датчик BH1750 обладает следующими характеристиками:

- напряжение питания – 2,4 – 3,6 В;
- ток потребления – 120 мкА;
- измеряемая длина волны – 560 нм;
- точность в режиме высокого разрешения – 1 Лк;
- точность в режиме низкого разрешения – 4 Лк;
- период измерения в режиме высокого разрешения – 120 мс;
- период измерения в режиме низкого разрешения – 16 мс;
- АЦП – 16 бит.

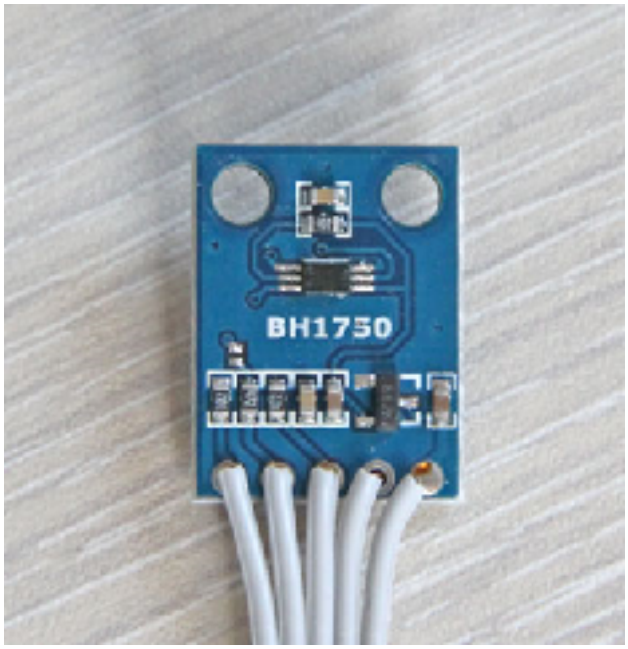


Рисунок 4.6 – Внешний вид датчика освещенности BH1750

4.1.3 Датчик частиц

Для отслеживания состояния атмосферы производственных условий в состав стенда был включен датчик пыли и частиц GP2Y1010AU0F от японского производителя электроники Sharp [31]. Датчик используется в промышленном оборудовании, в роботах-пылесосах, в системах пожарной сигнализации и др. Внешний вид датчика изображен на рисунке 4.6. Для передачи данных в ЭВМ используется шина I2C. Технические характеристики датчика GP2Y1010AU0F:

- напряжение питания: DC5 ± 2 В;
- ток потребления – 120 мкА;
- чувствительность: 0.5 В/(0.1 мг/м3), напряжение при чистом воздухе - 0.9 В;
- рабочая температура: -10 ~ 65;
- размер: 46 мм × 30 мм × 17.6 мм.

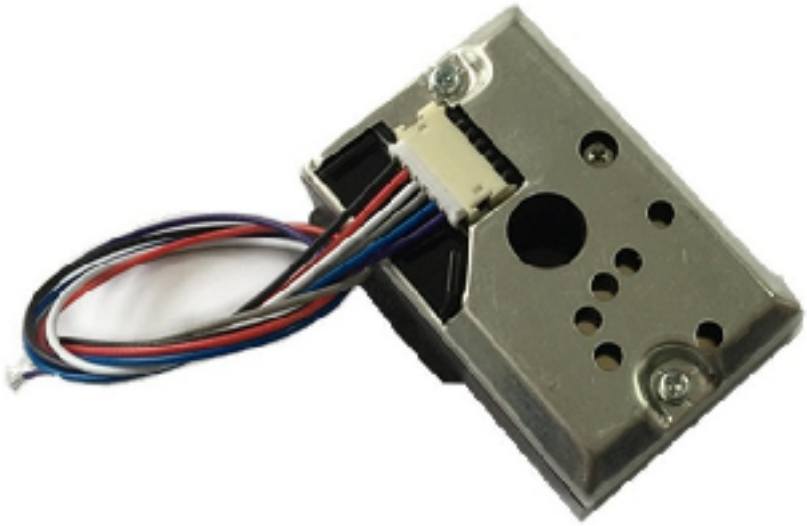


Рисунок 4.7 – Внешний вид датчика пыли и частиц GP2Y1010AU0F

4.1.4 Датчик расстояния

В качестве датчика расстояния был выбран цифровой датчик AP3216. Модуль имеет очень высокие показатели точности, температурную компенсацию, защиту от помех и мерцания, может работать за затемненным стеклом. Внешний вид модуля измерения расстояния изображен на рисунке 4.7. Характеристики датчика AP3216:

- интерфейс: I2C интерфейс (до 400k ГЦ);
- выбор режима: ALS, PS + IR;
- рабочее напряжение: 2,4 - 3.6VDC;
- рабочий ток: 700uA макс.

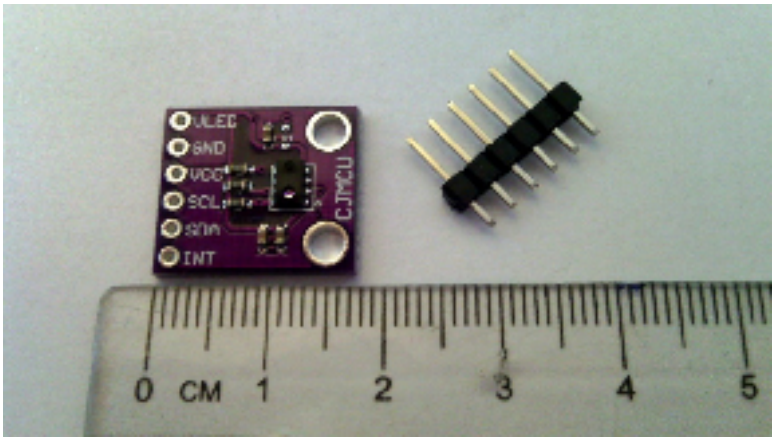


Рисунок 4.8 – Датчик AP3216

4.1.5 IP-камера

Для получения изображения и его последующего анализа в состав испытательного стенда входит IP-камера с объективом фиксированного типа. Под IP-камерой понимают цифровую видеокамеру, особенностью которой является передача видеопотока в цифровом формате по сети Ethernet и TokenRing, использующей протокол IP. Как правило, перед передачей, полученное с матрицы изображение сжимается с помощью покадровых (MJPG) или потоковых (MPEG-4, H.264) методов видеосжатия. Специализированные IP-камеры чаще осуществляют передачу видео в несжатом виде.

В качестве протокола транспортного уровня в IP-камерах могут использоваться протоколы: TCP, UDP и другие протоколы транспортного уровня модели OSI. Благодаря тому, что в IP камерах нет необходимости передавать аналоговый сигнал в формате PAL или NTSC, в IP-камерах могут использоваться большие разрешения, включая мегапиксельные. Типичное разрешение для сетевых камер: 640x480 точек. Существуют камеры с мегапиксельными разрешениями: 1280x1024, 1600x1200 и более высокими.

Моделью камеры используемой в составе стенда является IP-камера ORIENT IP-36-720. Внешний вид камеры можно наблюдать на рисунке 4.7. Характеристики IP-камеры:

- тип матрицы: CCD;

- число пикселей матрицы: 1 мп;
- фокусное расстояние: 3.6 мм;
- максимальное разрешение: 1280x720 пикселей;
- стандарт степени защиты: IP66.

Рисунок 4.9 – IP-камера ORIENT IP-36-720

4.2 Описание программного обеспечения стенда

Для решения поставленной задачи был использованы следующие программные продукты.

1) Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в

функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты). Эти свойства языка программирования облегчили задачу создания объектов для алгоритмов генетического программирования. Для разработки ПО использовалась версия языка программирования Python 3.6.

2) Библиотека OpenCV – библиотека алгоритмов обработки и анализа изображений, обзор используемых алгоритмов библиотеки проведен во второй главе данной работы.

3) Библиотека DEAP – библиотека распределенных эволюционных алгоритмов в Python. Она обеспечивает основу для работы с генетическими алгоритмами.

Разработанное ПО можно разбить на три части: описание деревьев решения, операции над деревьями решений (создание и мутация) и генерация системы обнаружения объекта. Листинг ПО представлен в приложении А. Структурная диаграмма программного обеспечения представлена в приложении Б.

4.2.1 Классы деревьев решений

Для реализации концепции дерева решения были разработаны следующие классы: FuncWrapper, FuncNode, SensorNode, ConstNode, ImageNode. Рассмотрим каждый из них подробно.

FuncWrapper. Обёрточный класс для функций. Каждый экземпляр этого класса имеет следующие поля:

- function – функция, лежащая в основе класса;
- childCount – количество аргументов, которое функция принимает;
- childDataTypes – типы данных аргументов (0 – логический, 1 – числовой, 2 – изображение);
- name – имя функции.

FuncNode. Функциональное звено. Единственный вид звена, который может иметь потомков. Каждый экземпляр этого класса имеет следующие поля:

- fw – экземпляр обёрточного класса функций;
- children – звенья-потомки.

Кроме того, этот класс, как и все остальные классы звеньев, имеет метод evaluate. В случае FuncNode, этот метод вычисляет ветку дерева, рекуррентно вызывая методы evaluate каждого потомка. Этот метод принимает на вход изображение, которое необходимо обработать, а также показания всех датчиков.

SensorNode. Звено, соответствующее показанию какого-либо датчика. Каждый экземпляр этого класса имеет поле index – номер датчика. При создании экземпляра этого класса index инициализируется указанным значением. Метод evaluate возвращает показание датчика с номером index (из массива, переданного в качестве второго аргумент метода).

ConstNode. Звено, соответствующее логической или числовой константе. Каждый экземпляр этого класса имеет поле value – значение константы. При создании экземпляра этого класса value инициализируется указанным значением. Метод evaluate возвращает value.

ImageNode. Звено, соответствующее обрабатываемому изображению. Метод evaluate возвращает обрабатываемое изображение (первый аргумент метода).

4.2.2 Операции над деревьями решений

Для создания случайного дерева используется функция *createTree*. В качестве аргументов она принимает максимальную глубину дерева и тип данных корневого звена (по умолчанию – изображение). Кроме того, для работы этой функции необходимо задать следующие переменные:

- createTree.nodeFactories – список из трёх элементов, каждый из которых является списком фабрик звеньев (объектов, создающих звенья) соответствующего типа данных (логический, числовой, изображение);
- createTree.nodeProbs – список из трёх элементов, каждый из которых является списком вероятностей выбора той или иной фабрики звеньев;
- createTree.leafFactories и createTree.leafProbs – аналогичные переменные, но для случая, когда максимальная глубина дерева достигнута (в этом случае нельзя использовать функциональные звенья).

Для создания звеньев разработаны фабрики звеньев. Все они имеют метод create(), который создаёт соответствующее звено.

FilterFactory. Класс, создающий звено FuncNode, фильтрующее изображение. Фильтр (а точнее экземпляр класса FuncWrapper, соответствующий фильтру) случайным образом выбирается из списка FilterFactory.filterWrapperList с равной вероятностью. В состав этого списка входят:

- gaussianBlurWrapper – размывание по Гауссу;
- medianBlurWrapper – медианный фильтр;
- dilateWrapper – дилатация;
- erodeWrapper – эрозия;
- equalizeHistWrapper – выравнивание гистограммы.

Метод create создаёт функциональное звено без указания потомков. Полагается, что потомки будут заданы позднее.

FuncFactory. Класс, создающий звено FuncNode с указанной функцией. При создании этой фабрики указывается экземпляр класса FuncWrapper. Метод create создаёт функциональное звено без указания потомков. Полагается, что потомки будут заданы позднее. В программе этот класс используется со следующими объектами класса FuncWrapper:

- ifWrapper – условное выражение;
- greaterThanWrapper – функция сравнения «больше».

SensorFactory. Класс, создающий звено SensorNode. При создании этой фабрики указывается число датчиков. Метод create создаёт звено, соответствующее датчику со случайным номером.

BoolConstFactory. Класс, создающий звено ConstNode с булевым значением. Значение с равной вероятностью равно 0 (ложь) или 1 (истина).

NumConstFactory. Класс, создающий звено ConstNode с числовым значением. Значение выбирается от 0 до 1 случайно.

ImageFactory. Класс, создающий звено ImageNode.

Для мутации дерева используется функция *mutateTree*. В качестве аргументов она принимает дерево, мутацию которого нужно произвести, вероятность мутации звена, максимальную глубину дерева и тип данных корневого звена (по умолчанию – изображение).

4.2.3 Генерация системы обнаружения объекта

Система обнаружения объекта генерируется в соответствии с методикой, описанной в главе 3. Рассмотрим необходимые для этого функции.

Distance. Функция расстояния между двумя точками. Используется для вычисления ρ_i в формуле (3.2).

distancePriv. Функция приведённой расстояния между двумя точками. Используется для вычисления $\rho_{i\ прив}$ в формуле (3.1).

fitnessFcn. Функция фитнеса. Вычисляется по формуле (3.1). На вход принимает объект, который должен иметь следующие поля:

- cameraFilter – процедура T_1 предобработки изображения с камеры;
- templateFilter – процедура T_2 предобработки шаблонного изображения;
- detectionParam – порог c в алгоритме обнаружения объекта.

Кроме того, для работы этой функции необходимо задать следующие переменные:

- fitnessFcn.sample – обучающая выборка; список элементов со следующими полями:
    1) cameraImage – изображение с камеры;
    2) sensors – показания датчиков в момент взятия изображения;
    3) objectPresent – наличие или отсутствие искомого объекта;
    4) pos – положение искомого объекта в пикселах (если отсутствует, то равно None);
- fitnessFcn.templateImage – шаблонное изображение;
- fitnessFcn.detect – процедура обнаружения объекта;
- fitnessFcn.distMax – параметр $\rho_{\max}$ в формуле (3.1);
- fitnessFcn.gamma – параметр γ_{un} в формуле (3.1);
- fitnessFcn.detectionErrorWeight – параметр w_{un} в формуле (3.1);
- fitnessFcn.falsePositivesWeight – параметр $w_{лп}$ в формуле (3.1);
- fitnessFcn.falseNegativesWeight – параметр $w_{ло}$ в формуле (3.1).

createIndividual. Функция, создающая случайную особь.

Mate. Функция, производящая потомка от двух родителей.

Mutate. Функция, производящая мутацию особи.

Main. Функция для проведения эксперимента. В ней необходимо указать:

- popSize – размер популяции;
- csProb – вероятность скрещивания;
- mutProb – вероятность мутации;
- genCount – число поколений.

4.3 Экспериментальные исследования

Для проведения испытаний разработанных алгоритмов требуется составить обучающую выборку изображений объекта распознавания. Для создания обучающей выборки используются средства самого стенда. В обучающую выборку вошли снимки метки ориентации автомобильного диска и

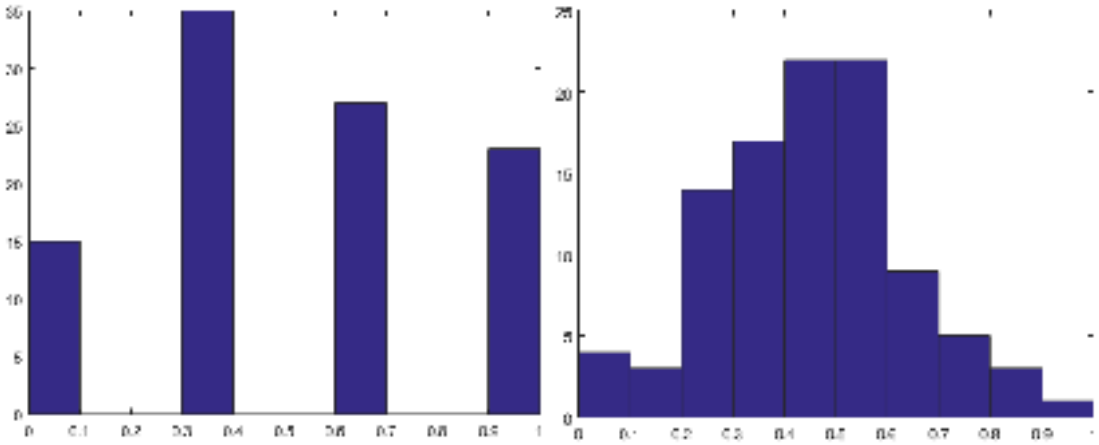
небольшая область реборды вокруг метки, показания датчиков частиц, расстояния и освещенности, записанные в текстовый файл.

4.4.1 Методика проведения эксперимента

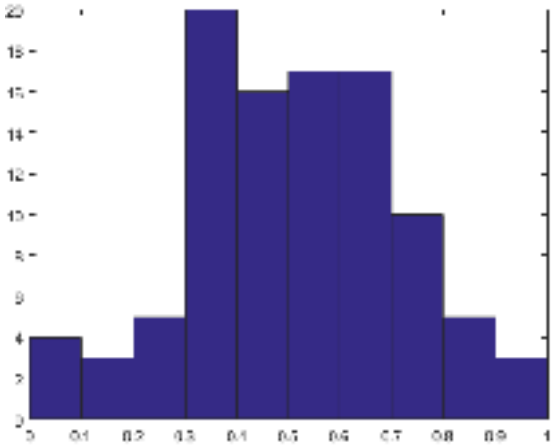
Описание составленной выборки:

- размер: 200 изображений;
- каждый элемент выборки состоит из следующих данных: изображение с камеры, показания трёх датчиков в момент взятия изображения, наличие или отсутствие метки на изображении, положение метки в пикселах (если она присутствует);
- приведённые показания датчиков варьируются от 0 до 1;
    - приведённое показание датчика расстояния, равное 0, соответствует наименьшему диаметру колеса (15 дюймов); значение 1 соответствует наибольшему диаметру (18 дюймов);
    - приведённое показание датчика освещения, равное 0, соответствует самому низкому освещению в выборке; значение 1 соответствует самому высокому;
    - приведённое показание датчика взвеси, равное 0, соответствует самому чистому воздуху в выборке; значение 1 соответствует самому загрязнённому.
- количество изображений с меткой – 100, без метки – 100; при (приблизительно) одних и тех же значениях датчиков берётся по два изображения одного и того же колеса – с меткой и без метки.

Распределение показаний датчиков в обучающей выборке показано на рисунке 4.8.



а) б)



в)

а) датчик расстояния; б) датчик освещенности; в) датчик частиц

Рисунок 4.8 – Гистограммы показаний датчиков

Матрица корреляции показаний трёх датчиков представлена в таблице 4.1.

Таблица 4.2 – Параметры экспериментов

	Расстояние	Освещение	Частицы
Расстояние	1.0000	0.0652	-0.0193
Освещение	0.0652	1.0000	-0.6853
Частицы	-0.0193	-0.6853	1.0000

Выборка делится на две равные части по 100 элементов: обучающая и тестовая выборки. Первая используется при генерации системы обнаружения; вторая используется для анализа качества работы системы. При составлении каждой части было взято по 50 изображений с метками на колёсах и по 50 изображений без метки на тех же 50 колёсах.

4.4.2 Проведение эксперимента и результаты анализа данных

С целью исследования влияния выбора алгоритма обнаружения объекта на изображении и параметров функции фитнеса на полученную систему, было проведено 4 эксперимента. Параметры, при которых были проведены эксперименты, представлены в таблице 4.2. Используемые в экспериментах параметры генетического алгоритма приведены в таблице 4.3.

Таблица 4.3 – Параметры экспериментов

№	Алг. Обнаружения	$N_{max\ глуб}$	$\rho_{\max}$	$\gamma_{ип}$	$w_{ип}$	$w_{лп}$	$w_{ло}$
1	Гомография точек	4	30	0.8	0.33	0.33	0.33
2		4			0.3	0.55	0.25
3		6			0.33	0.33	0.33
4		2					
5	Сопоставл. Шаблона	4					

Таблица 4.4 – Параметры генетического алгоритма

Параметр	**Значение**
Размер поколения	30
Число поколений	50
Генерация первого поколения	Алгоритм «создать дерево» запускается дважды для каждой особи: для создания деревьев T_1 и T_2; затем случайно генерируется параметр c алгоритма обнаружения объекта из диапазона [0, 1].

Окончание таблицы 4.3

Селекция	Выживают 50% наиболее приспособленных особей
Выбор родителей	Оба родителя выбираются случайно, каждая особь популяции имеет равные шансы быть выбранной
Скрещивание	С равной вероятностью извлекаются один за другим соответствующие элементы генотипов родителей A и B: $s_A = \{T_{A1}, T_{A2}, c_A\}$ и $s_B = \{T_{B1}, T_{B2}, c_B\}$.
Вероятность мутации	10%
Мутация	Для деревьев T1 и T2 выполняется алгоритм «Мутация_дерева»; к параметру c прибавляется случайное число с нормальным распределением $N(0, 0.2)$. Если результат окажется меньше 0, то c заменяется значением 0; если результат превысит 1, то c заменяется значением 1.

Результаты первого эксперимента в виде лучших особей показаны на рисунках 4.9 и 4.10.

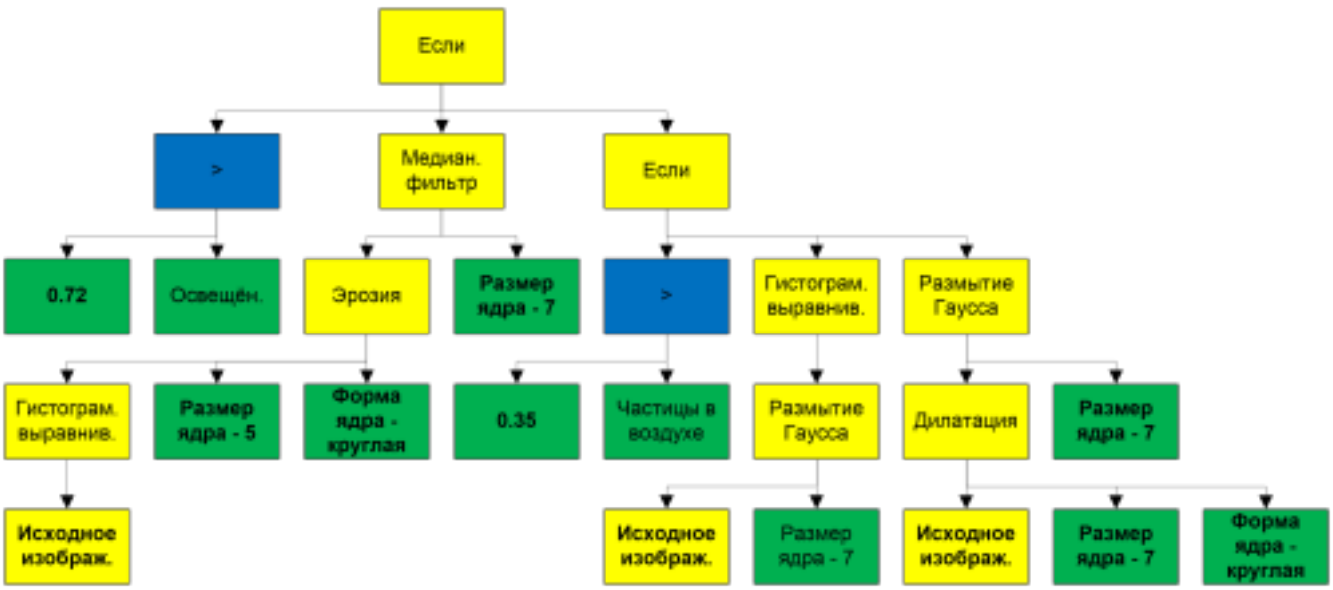


Рисунок 4.9 – Дерево T1 предобработки изображения с камеры

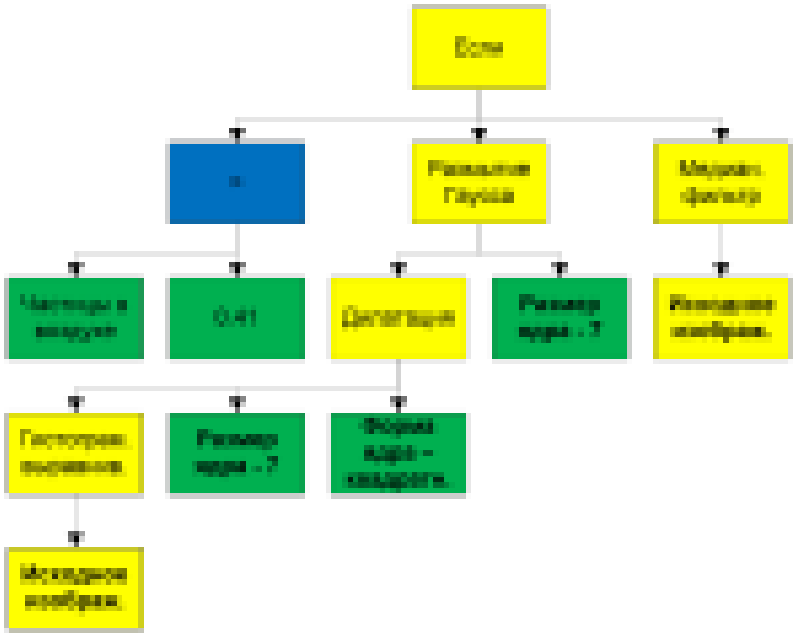


Рисунок 4.10 – Дерево T2 предобработки шаблонного изображения

Качество работы системы оценивается по трём признакам: процент ложноположительных случаев, процент ложноотрицательных случаев и средняя приведённая погрешность обнаружения в истинно-положительных случаях. Эти данные приведены в таблице 4.4.

Таблица 4.4 – Качество работы систем обнаружения

№	Обучающая выборка			Тестовая выборка			c	$N_{глуб}$
	% лож. Полож.	% лож. Отриц.	Сред. Прив. Погреш. Обнаружен.	% лож. Полож.	% лож. Отриц.	Сред. Прив. Погреш. Обнаружен.		
1	0	1	0.4304	4	3	0.6352	0.8293	4
2	1	3	0.5032	0	7	0.6820	0.6934	4
3	2	1	0.3995	4	4	0.6104	0.8511	4
4	5	6	0.6413	11	13	0.7231	0.7533	2
5	3	2	0.5109	6	8	0.6689	0.4490	4

4.5 Выводы по главе

Анализ результатов показал, что в первом эксперименте в обучающей выборке ложноотрицательные ошибки детектирования составили 1%. Во втором эксперименте на тестовой выборке показано, что за счёт назначения разных весов удалось избавиться от ЛП-ошибок, но процент ЛО-ошибок увеличился. Третий эксперимент показал, что увеличение максимальной глубины до шести уровней дерева не улучшило результат, однако, из четвёртого эксперимента ясно, что максимальная глубина равная двум уровням – явно недостаточна. В пятом эксперименте были повторно введены параметры первого эксперимента, но совместно с другим алгоритмом детектирования. Из результатов эксперимента следует, что метод сравнения гомографии точек лучше справляется с задачей. Стоит также отметить, что результаты работы системы на тестовой выборке хуже, чем на обучающей, что вполне ожидаемо.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена исследованию методов и средств самоадаптивного управления встраиваемыми системами, а также разработке метода и программного обеспечения на основе эволюционных методов и генетического программирования. Для достижения цели, заявленной в работе, был осуществлён обзор и анализ существующих методов разработки интеллектуальных систем для программного обеспечения ВС. На основании проведённого анализа были обозначены основные задачи, определены методы их решения. Была разработана схема работы встраиваемой системы, для которой было выбрано соответствующее оборудование и разработан алгоритм функционирования. Разработанная система была внедрена в эксплуатацию на действующем предприятии ООО «K&K» г. Красноярск, при производстве автомобильных дисков на автоматизированной линии механообработки. Результаты диссертационной работы послужат материалом для дальнейших исследований применения эволюционных методов для разработки самоадаптивных интеллектуальных систем управления ВС.

СПИСОК СОКРАЩЕНИЙ

АПК аппаратно-программный комплекс

ПО программное обеспечение

ВС встраиваемая система

ЭП эволюционное программирование

ГА генетический алгоритм

ГП генетическое программирование

ПЛК промышленный контроллер

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Букатов, А. А. Программирование многопроцессорных вычислительных систем : учебное пособие / А. А. Букатов, В. Н. Дацюк, А. И. Жегуло. – Ростов-на-Дону : Изд-во ООО «ЦВВР», 2003. – 208 с.
2. Гэри, М. Вычислительные машины и труднoразрешимые задачи : пер. с англ. / М. Гэри, Д. Джонсон. – Москва : Мир, 1982. – 416 с.
3. Vahid F. Hardware/Software Codesign / F. Vahid // Proceedings of International Workshop, Colorado : Estes Park,2002. 2002. - P. 1-6.
4. Алпатов. Б. А. Системы автоматического обнаружения и сопровождения объектов. Обработка изображений и управление : науч. изд. / Б. А. Алпатов. – Москва : Радиотехника, 2008. – 176 с.
5. Обработка и анализ цифровых изображений с примерами на LabVIEW IMAQ Vision : учеб. пособие / Ю. В. Визильтер, С. Ю. Желтов, В. А. Князь, А. Н. Ходарев, А. В. Моржин. – Москва : ДМК Пресс, 2007. – 464 с.
6. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – Москва : Техносфера, 2006. – 1072 с.
7. AForge.NET Framework [Электронный ресурс] // Google Code – Режим доступа: http://code.google.com/p/aforge/
8. VXL Framework [Электронный ресурс] // Google Code – Режим доступа: http://code.google.com/p/vxl/
9. Bradsky G. Learning OpenCV. Computer Vision with the OpenCV Library / G. Bradsky, A. Kaebler. – New York : O'RELLY, 2008. – 253 с.
10. Евдокимов, Ю. К. Система технического зрения для геометрических микроизмерений / Ю. К. Евдокимов, М. И. Николаев // Вестник КГТУ. — 2006. — Выпуск 4(44). — С. 21 – 23.
11. Davies E. R. Machine Vision: Theory, Algorithms, Practicalities : Academic Press / E. R. Davies. – San Diego : Elsevier, 1997. – 750 с.
12. Проненко В. Д. Оценка эффективности разработанного программного

комплекса, использующего-поиск визуальной информации в базе данных при принятии решений. / В. Д. Проненко // Вестник РГРТУ – 2005. – № 4. – С. 31–34.

13. Платунов А. Е. Архитектурная модель цифровых вычислительных систем для встроенных применений. // Изв. вузов. Приборостроение. Т.44. №3. 2001. С.8 – 15.

14. Платунов А. Е. Заказные вычислительные платформы информационно-управляющих систем. Презентация компании ЛМТ. // Электронные компоненты. № 12. 2005. С. 42.

15. Рыбаков А. П. Современные открытые международные стандарты для построения интегрированных измерительных и управляющих систем реального времени // Мир компьютерной автоматизации. 1995. - №1. – С. 5 – 12.

16. Blume H. Parallel Predictive Motion Estimation using Object Recognition Methods / H. Blume, A. Amer // Proceedings of the European Workshop and Exhibition on Image Format Conversion and Transcoding. – vol. 9. – no. 2. – PP. 813-631.

17. Горбань А. Н. Обучение нейронных сетей / А. Н. Горбань. – Москва : СП «ПараГраф», 1990. 160с.

18. Mari M. Interactive object detection using a fuzzy image segmentation approach. / M. Mari // Machine GRAPHICS & VISION 1998. – vol. 7. – no. 4. - PP. 765-780.

19. Bishop C. M. Neural Networks and Pattern Recognition / C. M. Bishop. – Oxford : Press, 1995. – 168 p.

20. Fogel L. J. Artificial Intelligence throw Simulated Evolution / J. L. Fogel, A. J. Owens. – New York : Wiley, 1966. – 265 p.

21. Mandavilli S. Adaptation in genetic algorithms / S. Mandavilli, L. M. Patnaik // In: Genetic algorithms for pattern recognition. CRC Press. – 1996. – P.45 – 64.

22. Скобцов Ю. А. Основы эволюционных вычислений / Ю. А. Скобцов. – Донецк : ДонНТУ, 2008. – 326с.

23. Eiben A. Adaptation in evolutionary computation: a survey / A. Eiben, R. Hinterding, Z. Michalevicz // Proceedings of IEEE international conference on evolutionary computation., New York : Piscataway, 1997 – P.65 – 69.

24. Фурман, Я. А. Введение в контурный анализ, приложения к обработке изображений и сигналов / Я. А. Фурман, А. В. Кревецкий, А. К. Передреев [и др.] ; под. общ. ред. Я. А. Фурмана. — Изд. 2-е, испр. – Москва : ФИЗМАТЛИТ, 2003. — 592 с.

25. Seeger U. R. Fast corner detection in grey-level images/ U. R. Seeger // Pattern Recogn. Lett., 1994, №15(7), pp.669 – 675.

26. Максимов Н. А. Алгоритмы цифровой обработки изображений : Учеб. Пособие / Н. А. Максимов, А. С. Василевский. – Москва : Изд-во МАИ, 1995, 31 с.

27. Lee C. H. Fast Motion Estimation Algorithm Based on the Block Sum Pyramid / C. H. Lee // IEEE Trans. Image Processing, vol. 6, pp. 1587 – 1591, Nov. 1997.

28. Николаев, М. И. Аппаратная реализация системы технического зрения для измерения геометрии микрообъектов Текст. / М. И. Николаев // Электронное приборостроение. Научно-практический сборник. Выпуск 3(44). — Казань : ЗАО «Новое знание», 2005. – С. 19 – 29.

29. Koza, John R. Genetic Programming: On Programming Computer by Means of Natural Selection and Genetics / John R. Koza. Cambridge, MA: The MIT Press, 1992.

30. Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning / D.E. Goldberg. Reading, Massachusetts: Addison-Wesley, 1989.

31. Compact Optical Dust Sensor [Электронный ресурс] // SparkFun – Режим доступа: https://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf

ПРИЛОЖЕНИЕ А

Код программы генетических операций

```
import cv2
import numpy as np
import random
from copy import deepcopy
from deap import base, creator
from deap import tools

class FuncWrapper:
def __init__(self, function, childDataTypes, name):
self.function = function
self.childCount = len(childDataTypes)
self.childDataTypes = childDataTypes
self.name = name

class FuncNode:
def __init__(self, fw, children = None):
self.fw = fw
self.children = children
def evaluate(self, image, sensors):
results = [n.evaluate(image, sensors) for n in self.children]
return self.fw.function(results)

class SensorNode:
def __init__(self, index):
self.index = index
def evaluate(self, image, sensors):
return sensors[self.index]

class ConstNode:
def __init__(self, value):
self.value = value
def evaluate(self, image, sensors):
return self.value

class ImageNode:
def evaluate(self, image, sensors):
return image

def createTree(maxDepth, atatype = 2):
if maxDepth == 0:
```

```
factories = createTree.leafFactories[atatype]
probs = createTree.leafProbs[atatype]
else:
factories = createTree.nodeFactories[atatype]
probs = createTree.nodeProbs[atatype]
factory = np.random.choice(factories, p = probs)
node = factory.create()
try:
children = [None] * node.fw.childCount
for I in range(node.fw.childCount):
children[i] = createTree(maxDepth – 1, node.fw.childDataTypes[i])
node.children = children
except AttributeError:
pass
return node

def mutateTree(tree, mutationProb, maxDepth, atatype = 2):
if np.random.random() < mutationProb:
return createTree(maxDepth, atatype)
try:
for I in range(tree.fw.childCount):
mutatedChildren[i] = mutateTree(tree.fw.children[i], mutationProb, maxDepth – 1,
tree.fw.childDataTypes[i])
return FuncNode(tree.fw, mutatedChildren)
except AttributeError:
pass
return deepcopy(tree)

ifWrapper = FuncWrapper(lambda args: args[1] if args[0] else args[2], [0, 2, 2], ‘if’)
greaterThanWrapper = FuncWrapper(lambda args: args[0] > args[1], [1, 1],
‘isGreater’)
gaussianBlurWrapper = FuncWrapper(lambda args: cv2.GaussianBlur(args[0], (20 *
args[1], 20 * args[1]), 0), [2, 1], ‘medianBlur’)
medianBlurWrapper = FuncWrapper(lambda args: cv2.medianBlur(args[0], 20 *
args[1]), [2, 1], ‘medianBlur’)
dilateWrapper = FuncWrapper(lambda args: cv2.dilate(args[0], 20 * args[1],
iterations = 1), [2, 1], ‘dilate’)
erodeWrapper = FuncWrapper(lambda args: cv2.erode(args[0], 20 * args[1],
iterations = 1), [2, 1], ‘erode’)
equalizeHistWrapper = FuncWrapper(lambda args: cv2.equalizeHist(args[0]), [2],
‘equalizeHist’)

class FilterFactory:
def create(self):
```

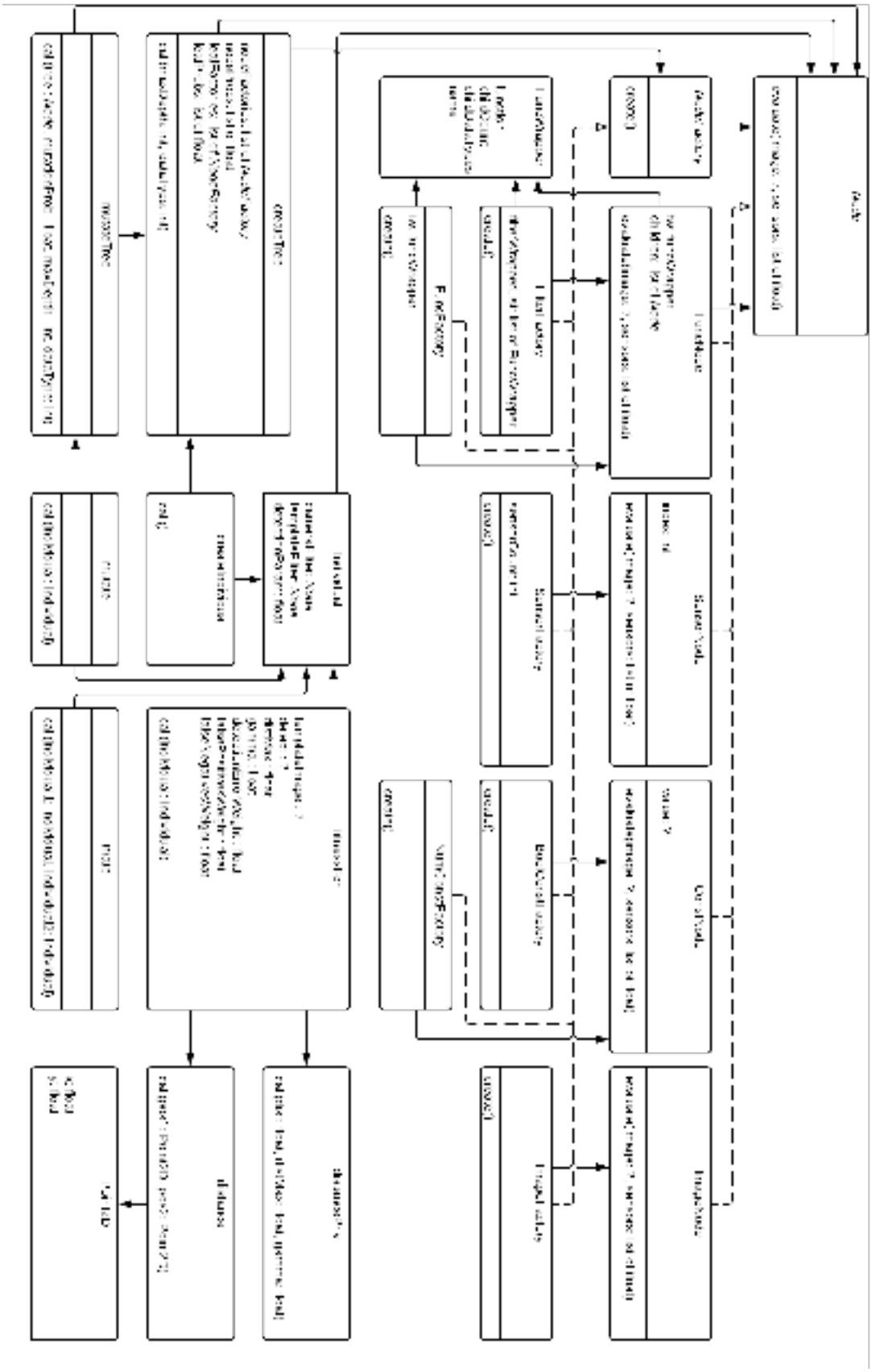
```
return FuncNode(np.random.choice(FilterFactory.filterWrapperList))

class FuncFactory:
def __init__(self, fw):
self.fw = fw
def create(self):
return FuncNode(self.fw)

class SensorFactory:
def __init__(self, sensorCount):
self.sensorCount = sensorCount
def create(self):
return SensorNode(np.random.randint(0, self.sensorCount))

class BoolConstFactory:
def create(self):
return ConstNode(np.random.choice([True, False]))

class NumConstFactory:
def create(self):
return ConstNode(np.random.random())

class ImageFactory:
def create(self):
return ImageNode()

sensorCount = 3

createTree.nodeFactories = [[FuncFactory(greaterThanWrapper),
BoolConstFactory()], [SensorFactory(sensorCount), NumConstFactory()],
[FuncFactory(ifWrapper), FilterFactory(), ImageFactory()]]
createTree.nodeProbs = [[0.5, 0.5], [0.5, 0.5], [0.2, 0.5, 0.3]]
createTree.leafFactories = [[BoolConstFactory()], [SensorFactory(sensorCount),
NumConstFactory()], [ImageFactory()]]
createTree.leafProbs = [[1], [0.5, 0.5], [1]]
FilterFactory.filterWrapperList = [gaussianBlurWrapper, medianBlurWrapper,
dilateWrapper, erodeWrapper, equalizeHistWrapper]

def distance(pos1, pos2):
return np.sqrt((pos1.x – pos2.x) ** 2 + (pos1.y – pos2.y) ** 2)

def distancePriv(dist, distMax, gamma):
return (min(dist, distMax) / distMax) ** gamma
```

```
def fitnessFcn(individual):
falsePositives = 0
falseNegatives = 0
detectionError = 0.0
for element in fitnessFcn.sample:
cameraFiltered = individual.cameraFilter.evaluate(element.cameraImage,
element.sensors)
templateFiltered = individual.templateFilter.evaluate(fitnessFcn.templateImage,
element.sensors)
detection = fitnessFcn.detect(cameraFiltered, templateFiltered,
individual.detectionParam)
if element.objectPresent:
if detection.objectPresent != element.objectPresent:
falseNegatives += 1
else:
dist = distance(detection.pos, element.pos)
distPriv = distancePriv(dist, fitnessFcn.distMax, fitnessFcn.gamma)
detectionError += distPriv
else:
if detection.objectPresent != element.objectPresent:
falsePositives += 1
return fitnessFcn.detectionErrorWeight * detectionError +
fitnessFcn.falsePositivesWeight * falsePositives + fitnessFcn.falseNegativesWeight *
falseNegatives;

fitnessFcn.templateImage = None
fitnessFcn.detect = None
fitnessFcn.distMax = 300
fitnessFcn.gamma = 1
fitnessFcn.detectionErrorWeight = 1
fitnessFcn.falsePositivesWeight = 1
fitnessFcn.falseNegativesWeight = 1

def createIndividual():
return [createTree(4), createTree(4), np.random.random()]

def mate(individual1, individual2):
for I in range(3):
if np.random.random() < 0.5:
individual1[i], individual2[i] = individual2[i], individual1[i]
return (individual1, individual2)

def mutate(individual):
individual[0] = mutateTree(individual[0])
```

```
individual[1] = mutateTree(individual[1])
individual[2] += np.random.normal(0, 0.1)
if individual[2] < 0:
individual[2] = 0
elif individual[2] > 1:
individual[2] = 1
return individual

creator.create(“FitnessMin”, base.Fitness, weights=(-1.0,))
creator.create(“Individual”, list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
toolbox.register(“individual”, createIndividual)
toolbox.register(“population”, tools.initRepeat, list, toolbox.individual)

toolbox.register(“mate”, mate)
toolbox.register(“mutate”, mutate)
toolbox.register(“select”, tools.selTournament, tournsize=3)
toolbox.register(“evaluate”, fitnessFcn)

def main():
popSize = 50
csProb, mutProb, genCount = 0.5, 0.2, 50
pop = toolbox.population(n = popSize)

fitnesses = map(toolbox.evaluate, pop)
for ind, fit in zip(pop, fitnesses):
ind.fitness.values = fit

for g in range(genCount):
offspring = toolbox.select(pop, len(pop))
offspring = map(toolbox.clone, offspring)

for child1, child2 in zip(offspring[::2], offspring[1::2]):
if random.random() < csProb:
toolbox.mate(child1, child2)
del child1.fitness.values
del child2.fitness.values

for mutant in offspring:
if random.random() < mutProb:
toolbox.mutate(mutant)
del mutant.fitness.values
```

ПРИЛОЖЕНИЕ Б

Структурная диаграмма программного обеспечения



ПРИЛОЖЕНИЕ В

Справка о внедрении

Общество с ограниченной ответственностью «ЭЛНИС»

Юрид. адрес: Россия, 660111, г. Красноярск, ул. Пограничников, 42

ИНН / КПП 2465209672 / 246501001 ОГРН 1082468027487

р/сч 40702810431280030712

КРАСНОЯРСКОЕ ОТДЕЛЕНИЕ № 8646 ПАО СБЕРБАНК г. КРАСНОЯРСК

БИК 040407627 к/сч 30101810800000000627

тел./факс: (391) 2564319, ooo@elnis.ru, www.elnis.ru, почт. адрес: 660059, г.Красноярск, а/я 3004

Справка о внедрении

Настоящим подтверждаю, что результаты диссертационного исследования Латышонок Н.А. на тему: «Алгоритмы и программно-аппаратные средства самоадаптивного управления встраиваемыми системами» обладают актуальностью, представляют практический интерес и были использованы при разработке и внедрению системы технического зрения распознавания моделей колес по дизайну лицевой поверхности на линиях механообработки № 11, № 12 и № 13 с изготовлением стенда для создания базы моделей колес на предприятии ООО «КиК»

Директор ООО "ЭЛНИС" С.В. Пост

"ЭЛНИС"