ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн
2
2020
Том 11

Рисунок к статье **М. В. Бобыря, А. А. Дородных, А. С. Якушева, В. А. Булатникова**
«АППАРАТНО-ПРОГРАММНЫЙ МЕХАТРОННЫЙ КОМПЛЕКС ДЛЯ ФИКСАЦИИ ПОДВИЖНЫХ ОБЪЕКТОВ»

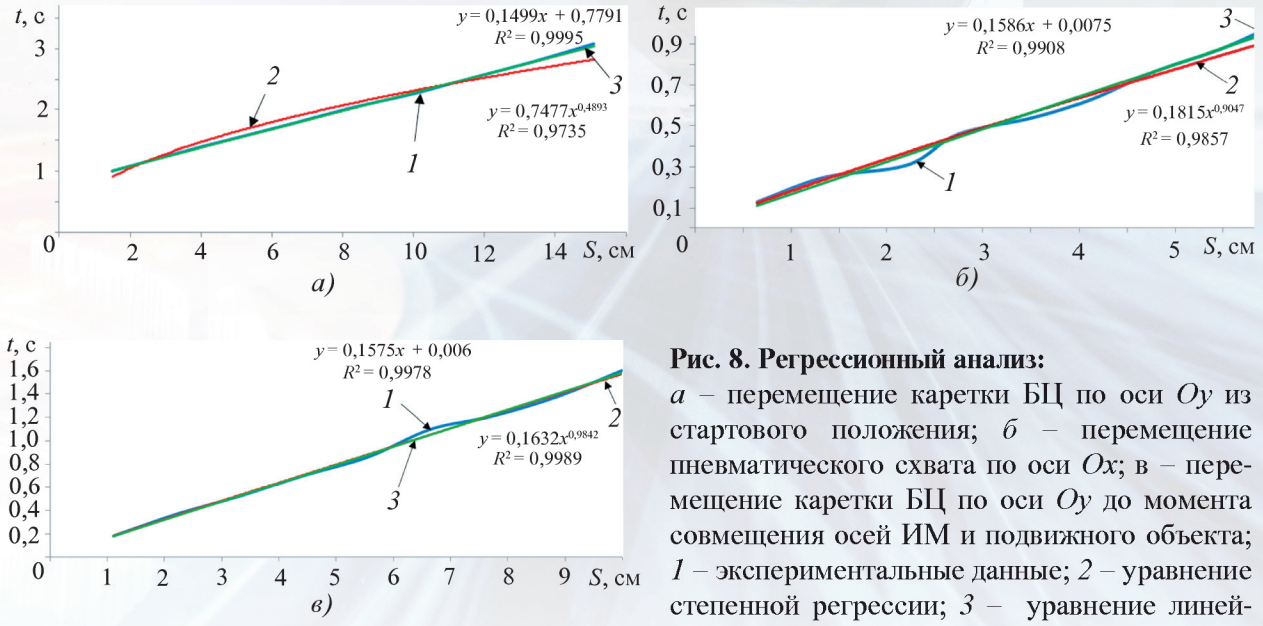

Рис. 8. Регрессионный анализ:
а – перемещение каретки БЦ по оси *Oy* из стартового положения; *б* – перемещение пневматического схвата по оси *Ox*; в – перемещение каретки БЦ по оси *Oy* до момента совмещения осей ИМ и подвижного объекта; *1* – экспериментальные данные; *2* – уравнение степенной регрессии; *3* – уравнение линейной регрессии

Рисунок к статье **Д. И. Читалова**
«О РАЗРАБОТКЕ МОДУЛЯ ДЛЯ РЕАЛИЗАЦИИ ДВИЖЕНИЯ И ТОПОЛОГИЧЕСКОГО ИЗМЕНЕНИЯ РАСЧЕТНЫХ СЕТОК И ЕГО ИНТЕГРАЦИИ В ГРАФИЧЕСКУЮ ОБОЛОЧКУ ДЛЯ ПЛАТФОРМЫ OpenFOAM»

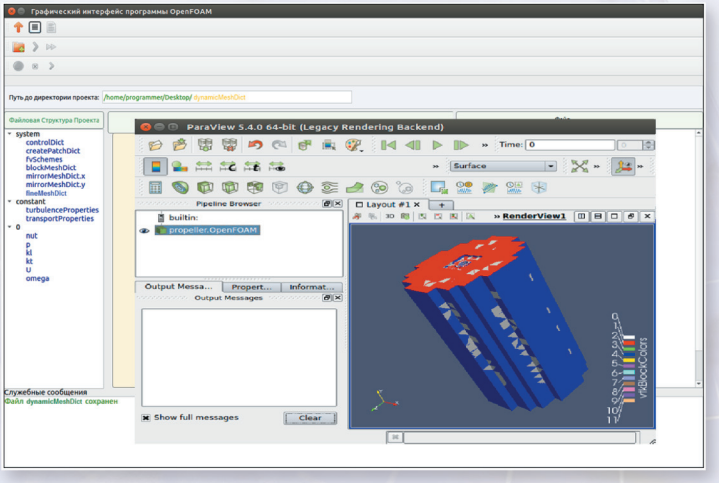

Рис. 3. Визуализация результатов работы утилиты moveDynamicMesh на примере одной из учебных задач платформы OpenFOAM

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 11
N 2
2020

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 338.364.2 DOI: 10.17587/prin.11.67-76

Д. С. Пащенко, канд. техн. наук, независимый консультант в области разработки программного обеспечения, denpas@rambler.ru, г. Москва

Отражение мировых практик программной инженерии и управления качеством программного обеспечения в отечественной IT-отрасли: результаты исследования по регионам России

Приведены обобщенные результаты двух авторских исследований 2017 и 2019 гг. об отражении современных мировых практик программной инженерии и управлении качеством программного обеспечения в отечественных IT-компаниях. В исследованиях приняли участие инженеры, аналитики, менеджеры проектов из всех федеральных округов РФ. Проведен соответствующий анализ и даны выводы о динамике основных процессов и схожести технологических и организационных сдвигов в российской программной инженерии в сравнении с мировыми тенденциями.

Ключевые слова: *программная инженерия, управление качеством программного обеспечения, организация команд разработки, управление проектами разработки*

Введение и постановка задачи

Развитие методологии программной инженерии в последние годы происходит стремительным образом. Этому способствует мировая тенденция цифровизации экономики, которая в последние годы активно развивается и в России. К уже давно сформированным трендам в области перехода к "гибким" и "легковесным" парадигмам разработки программного обеспечения (ПО) присоединяются новые технологии управления версионностью и обеспечением непрерывной интеграции и поставки релизов ПО. Существенное значение приобретают новые формы организации команд разработки, включая географически распределенные команды, работу вне офисов и мобильные средства планирования и коммуникаций.

В условиях глобальной конкурентной борьбы отрасль информационных технологий развивается чрезвычайно быстро, при этом разработка ПО как часть отрасли переживает существенное смещение в технологических и организационных подходах. Методы и технологии программной инженерии, а также инструментальные средства обеспечения качества, своевременной поставки и интеграции ПО претерпели серьезные изменения за последние 10 лет.

Очевидные тренды перехода к "гибким" и "легковесным" технологиям были существенно дополнены различными артефактами и практиками (результатами практического применения) из других парадигм — бережливого управления, синергетических и географически распределенных организаций.

Кроме того, следует отметить активное развитие перечисленных далее трендов.

- Mobile-First (сначала мобильные) [1] — разработка информационного ресурса (системы, сайта) ведется из ожиданий, что пользователи будут использовать преимущественно мобильные браузеры (смартфоны, планшеты) для взаимодействия с системой.
- Test Driven Development (разработка, управляемая тестированием) [2] — разработка ПО, в которой модель качества продукта, в том числе автотесты, симуляторы окружения, unit-тесты создаются до разработки самого программного продукта и определяют как его уровень качества, так и соответствие всем требованиям заказчика, пользователей и разработчиков.
- DevOps (английский акроним разработки и операционной деятельности в программной инженерии) [3] — методология активного и формализованного взаимодействия между разработчиками ПО и инженерами, осуществляющими операции по развертыванию, обновлению, непрерывному тестированию релизов ПО, поддержке эксплуатации и помощи пользователям, созданию необходимых тестовых, промышленных и миграционных сред эксплуатации.

Эти тренды существенно влияют на процессы разработки ПО и практический подбор инструментальных средств для командной работы и реализации современного ПО. Не менее значимым является влияние различных факторов трудового рынка, которые определяют необходимость конкурентной борьбы IT-компаний за лучших специалистов на всех развитых рынках [4].

Однако следует отметить, что глобальное развитие данных трендов в мире и даже в условиях одного локального рынка происходит однозначно неравномерно. Особенно очевидна эта неравномерность по отраслям прикладной автоматизации и цифровизации. Так, отрасли с достаточными инвестицион-

ными ресурсами и высоким уровнем конкуренции уже перешли к современным технологиям и методологиям разработки ПО: в наиболее прогрессивных направлениях (технологические стартапы, финтех, заказная разработка ПО, лидеры интернет-сервисов и электронной коммерции) ведущие мировые тенденции уже стали де-факто стандартами. Близки к такому положению и крупнейшие игроки финансового и страхового рынков, завершающие цифровую трансформацию и построившие современные внутренние подразделения разработки ПО. Значительный рынок государственных информационных услуг и программных сервисов в России живет в парадигме собственных тенденций, в которых сочетаются стремление к последним достижениям в области технологий с открытым исходным кодом и новаторские подходы в области оказания государственных услуг с технологическим консерватизмом и декларативным импортозамещением [5].

Технологические инновации в современной программной инженерии носят гибкий и при этом завершенный характер. За каждой идеей и практическим корпоративным опытом в разработке современного ПО стоит автоматизирующий данную идею набор инструментальных средств (далее по тексту — инструментов) [6]. Более того, такие лидеры рынка автоматизации разработки ПО, как IBM, Oracle и Google создают целые линейки соответствующих программных продуктов, поддерживающих идеологию, процессы и артефакты для популярных методологий [7, 8]. Рост мировой популярности подходов в области обеспечения процессов версионного хранения, сборки и непрерывной доставки и интеграции ПО (*Continues Integration / Continues Delivery,* CI/CD) сопровождается популяризацией соответствующих средств автоматизации. Такие инструменты, как GIT, GITLab, Jetkins, Docker вслед за мировой популярностью обрели высокую скорость получения новых функциональных возможностей и целый набор рекомендованных моделей управления версиями и релизами программных продуктов.

В настоящей работе собраны и проанализированы отдельные обобщенные результаты двух авторских исследований по федеральным округам РФ, проведенных в начале 2017 г. и в конце 2019 г. и охвативших соответственно 79 и 68 инженеров, аналитиков, руководителей проектов по разработке ПО. Данные исследования включали сбор и анализ информации об актуальности мировых тенденций в разработке и проектировании ПО в практике отечественных команд. Полные результаты первого исследования 2017 г. доступны в работах [5, 9]. Основными задачами являются оценка востребованности мировых практик программной инженерии в командах разработки ПО из России и анализ соответствия российского рынка разработки ПО современным технологическим требованиям.

Методика исследований

Исследования были проведены в марте—апреле 2017 г. и в ноябре—декабре 2019 г. по схожей методике, включающей два этапа. Состав обеих панелей отбирался через сеть профессиональных контактов автора в отрасли или рекомендации квалифицированных коллег с обязательным соблюдением следующих условий:

- доказанный опыт каждого эксперта в разработке ПО в течение последних лет;
- возраст старше 20 лет;
- любую компанию могут представлять не более двух сотрудников с различными ролями (инженер, разработчик, аналитик, руководитель проекта) и работающих в настоящий момент в разных проектах;
- опыт эксперта в разработке ПО релевантен хотя бы одному из регионов (федеральных округов) РФ.

На первом этапе каждый эксперт отвечал на набор вопросов в анкете Google.Form с заранее определенными вариантами ответа по слеудующим разделам исследования:

1) популярность инструментов, технологий и паттернов в разработке программного обеспечения;

2) востребованность подходов к организации производственных процессов;

3) подходы к проектированию информационных систем;

4) локальные российские тренды (импортозамещение, защита данных и т. п.).

Далее ответы в автоматизированном режиме средствами Google.Form обобщались в документ с результатами исследований и данный документ отправлялся каждому эксперту.

Часть экспертов на втором этапе отправили свои замечания, возражения и комментарии, которые были добавлены в итоговую версию результатов исследования. В настоящей работе приведены как итоговое распределение мнений экспертов по вариантам ответов, так и набор дополнительных замечаний, поясняющих отдельные мнения о востребованности тех или иных подходов, инструментов и технологий.

Рассмотрим характеристики двух панелей экспертов, участвовавших в исследованиях в 2017 и 2019 гг. в разрезах возраста и профессионального опыта экспертов и регионов получения такого опыта в разработке ПО. В исследовании 1 от апреля 2017 г. участвовали 79 экспертов, а в исследовании 2 от декабря 2019 г. — 68 экспертов. Персональный состав участников обеих панелей совпадал примерно на 40 %.

В обоих исследованиях значительная часть экспертов занимается разработкой программного обеспечения более 10 лет (табл. 1), а значит, в течение их карьеры в той или иной степени развивались основные доминирующие в мире тренды:

- сдвиг парадигм разработки к "гибким" методологиям;
- появление и бурное развитие мобильного ПО;
- закрепление доминирующего тренда web-разработки;
- активное развитие концепций трехзвенной, модульной, микросервисной типов архитектуры ПО.

Отметим также, что значительный процент экспертов в обоих исследованиях находится в возрасте от 30 до 39 лет (табл. 2). Данный возраст является самым плодотворным в IT-профессиях и

Таблица 1

Сколько лет Вы занимаетесь профессиональной разработкой ПО, соответствующими проектами и командами?

Вариант ответа, лет	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
1...3	2,5	3,0
3...6	32,9	13,2
6...10	20,3	23,5
Более 10	44,3	60,3

Таблица 2

Определите свою возрастную группу

Вариант ответа, лет	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
20...29	41,8	22
30...39	53,2	61,8
40...49	5	14,7
Старше 50	0	1,5

сопряжен с максимальной работоспособностью и профессиональными успехами.

В табл. 3 приведены распределения экспертов по профилям опыта, связанным с направлением разработки ПО. Среди профилей опыта представлены:

- разработка для собственных нужд компании (собственная разработка);
- разработка в рамках проектов по системной интеграции;
- разработка программных продуктов для рынка поставщиком (вендором — ISV);
- разработка ПО на заказ по уникальным требованиям клиента.

В табл. 4 приведены данные о региональном распределении экспертов по столицам и федеральным округам. В обоих исследованиях более трети экспертов представили Москву, что подтверждает действительно высокую концентрацию IT-компаний в столице.

Значимый акцент в обоих исследованиях сделан на двух направлениях, анализ которых приводится в настоящей работе:

- современные тенденции в области программной инженерии и организации команд разработки ПО в России;
- отражение мировых тенденций в области обеспечения качества ПО и надежности его своевременной поставки и интеграции в практике отечественных команд.

В завершении статьи приведены общие выводы о динамике процессов и актуальности мировых трендов в отечественной IT-отрасли.

Современные тенденции в области программной инженерии и организации команд: результаты исследований

Традиционные итерационные парадигмы [10], основанные на подходах RUP (*Rational Unified Process*) и MSF (*Microsoft Solution Framwork*), начали утрачивать мировое доминирующее положение в начале века. Примерно 10 лет назад переход к "гибким" и "легковесным" моделям в России стал не только темой модных конференций, но и целью реальных проектов внутри IT-компаний. В последние пять лет многие крупные корпорации в ходе реорганизации своих IT-подразделений начали использовать "гибкие" и "гибридные" парадигмы разработки ПО. Результаты авторских исследований об использовании "гибких" методов в разработке ПО (Agile, Scrum, XP) приведены в табл. 5.

Результаты обоих исследований указывают на то, что отдельные методики из гибкой/гибридной разработки закрепились в практике российских IT-компаний. Из частных мнений и замечаний экспертов, представленных на вторых этапах исследований, речь идет прежде всего о следующих инструментах и практиках:

1) регулярные короткие встречи по мониторингу прогресса (*Stand Up Meeting*);

2) разбиение разработки на короткие итерации и регулярная поставка результатов разработки в итерации на тестирование (в продуктивную среду);

3) планирование работ в условных единицах, не выраженных напрямую в некотором временном про-

Таблица 3

Представленный Вами опыт последних 2—3 лет больше всего относится..?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
К разработке для собственных нужд компании (*in-house*)	15,2	19,1
К проектам по системной интеграции (*system integrator*)	11,4	10,3
К разработке ПО (сервиса, технологии) независимым вендором (ISV)	36,7	27,9
К разработке ПО на заказ (включая *outsourcing*)	36,7	42,7

Таблица 4

Определите регион проживания (столицу, федеральный округ), в котором получен представленный опыт

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Москва	34,2	36,8
Санкт-Петербург и Северо-Западный ФО	10,1	8,7
Центральный ФО (без Москвы)	6,3	11,8
Южный и Северо-Кавказский ФО	7,6	7,4
Приволжский ФО	12,7	7,4
Уральский ФО	5,1	2,9
Сибирский ФО	21,5	11,8
Дальневосточный ФО	2,5	13,2

межутке выполнения задачи (человеко-дне, человеко-часе и т. п.);

4) совмещение ролей руководителя проекта и аналитика в роли владельца продукта (*Product Owner*).

Во многих японских и американских IT-компаниях "гибкие" и "легковесные" практики разработки ПО органически включили в себя элементы "бережливых" подходов (*Lean Software Development*) [11]. "Бережливость" в разработке ПО связана с акцентом команды на трату ресурсов преимущественно на создание самого программного продукта и минимизацию издержек в этом производственном процессе. Такие подходы включают: устранение потерь; акцент на обучение; целостное видение задач и их решений; использование только свершившихся фактов в анализе операционных и тактических задач. Современные "бережливые" парадигмы разработки ПО в чем-то идейно схожи с аналогичными концепциями автомобильного производства XX века в Японии (Toyota, Nissan, Honda): минимум необходимых усилий и рисков, минимальные сроки хранения незавершенной продукции, постоянное совершенствование производственных практик. Востребованность "бережливых" подходов в практике российских команд-разработчиков по результатам двух исследований в разные годы приведена в табл. 6.

Практически единственной популярной Lean-практикой для российских команд, опыт которых изучался в обоих исследованиях, стало использование Kanban-доски для мониторинга исполнения задач проекта. При этом часть команд использует электронные Kanban-доски (например, Trello или Jira Kanban), а часть команд — бумажные стикеры и пластиковые доски в комнатах разработчиков, наглядно мотивируя инженеров к скорейшему выполнению итерации разработки.

IT-компании в своем большинстве — это проектно-ориентированные организации, в которых должны быть закреплены успешные и формализованные практики проектного управления. Вне зависимости от того, как именно построена формализованная модель производства внутри каждого проекта компании, централизованный и регулярный контроль хода проектов разработки ПО должен влиять на их успешное выполнение. При этом организация

Таблица 5

В большей части компаний Вашего региона активно используются "гибкие" или "гибридные" методологии, подходы к разработке ПО и управлению проектами обновляются и становятся более "легковесными"?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
"Гибкая"/"гибридная" разработка используется как основная производственная практика	35,4	32,3
Какая-то часть методик "гибкой"/"гибридной" разработки используется в производстве ПО	49,4	55,9
Используются классические водопадные и итерационные модели (RUP, MSF, etc)	6,3	1,5
Используется собственная смесь подходов, но ничего "гибкого" в ней нет	8,9	10,3

Таблица 6

Отношение к использованию "бережливых" практик (*Lean Software Development*)

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
"Бережливые" практики используются в производстве ПО в большом объеме	6,4	4,4
Часть "бережливых" практик является составляющей процессов производства ПО	46,8	51,5
"Бережливое" производство ПО как подход/философия не используется совсем	46,8	44,1

Таблица 7

Отмечаете ли Вы рост востребованности централизованного и регулярного контроля проектов/команд и создания официальных подразделений для этого?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Да, все больше команд/проектов централизовано контролируются на регулярной основе	17,7	17,6
Да, тенденция очевидна, но без создания официального подразделения	21,5	39,7
Нет, такой тенденции не наблюдаю	51,9	32,4
Я вижу децентрализацию в проектном управлении	8,9	10,3

офиса управления проектами, лаборатории проектов, производственного центра управления, проектного офиса — это наиболее простой способ организации регулярного контроля проектов разработки ПО [12]. Рассмотрим востребованность данного подхода в российских командах (табл. 7).

Данные табл. 7 указывают, что формализация регулярного контроля проектов/команд, разрабатывающих ПО, с течением времени становится более востребованной. Однако идея создания специального организационного подразделения для этой цели и сегодня не обретает должного уровня востребованности. В этом направлении у российских команд есть пространство для роста конкурентных возможностей.

Еще одна мировая тенденция в области организации современных процессов разработки ПО — это использование географически распределенных команд. Такой подход означает, что разработкой одного сервиса или программного продукта занимается команда, состоящая из сотрудников разных офисов. С одной стороны, это позволяет существенно экономить фонд оплаты труда и использовать модели 24-часовой разработки ПО [13]. С другой стороны, такой подход заставляет изменить как коммуникации в команде, так и модели постановки (контроля) задач и совместной релизной работы. Востребованность использования географически распределенных команд в российской практике представлена в табл. 8.

Таким образом, опыт российских команд в целом согласуется с данной мировой тенденцией. Среди наиболее весомых причин использования географически распределенных команд эксперты выделили:

- экономию фонда оплаты труда;
- возможность привлекать лучших специалистов на нерегулярные работы в критических точках проекта;
- возможности быстрой адаптации глобальных программных продуктов к различным рынкам при поступлении требований и предложений, а также обратной связи от клиентов из различных географических регионов.

Кроме географически распределенных команд следует выделить схожую тенденцию организации труда — это работа членов команд разработки вне офисов компании. В мировой практике все чаще инженеры команд разработки ПО работают вне офиса, что связано с новыми подходами в организации командной работы, а именно экономией времени сотрудников на поездки в офис; предоставлением специальных преференций лучшим работникам; заботой об их семейном благополучии. В авторских исследованиях изучалась востребованность данного подхода в отечественных IT-компаниях (табл. 9).

Данная тенденция в достаточной степени востребована в опыте российских IT-компаний. Следует предположить, что с прогрессом средств автоматизации и удаленной коммуникации данный тренд будет заметен еще сильнее. Так, современные мобильные инструменты, включая мессенджеры, мобильные планировщики, групповые чаты, в мировой практике стали основой совместной работы и постоянной коммуникации членов команд разработки ПО. В табл. 10 отображена востребованность данной тенденции в отечественном опыте.

Безусловно, мобильные инструменты для рабочих коммуникаций — это удобно. Во многом такой подход согласуется с ростом популярности географически

Таблица 8

Географически распределенные команды стали стандартом отрасли и используются командами Вашего региона?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Согласен, почти все проекты разрабатываются географически распределенными командами	30,4	36,8
Согласен с тенденцией в отношении какой-то части проектов/направлений бизнеса	57	52,9
Не согласен, знакомые команды традиционно работают в одном офисе	12,6	8,8
Затрудняюсь ответить	0	1,5

Таблица 9

Отмечаете ли Вы, что все больше членов проектных команд работают вне офисов компании в силу изменения подходов к организации труда (в сторону улучшения трудовых прав и возможностей инженеров)?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Да, подходы к организации команд и бизнеса меняются	39,2	42,6
Да, но у этого есть другие объективные причины (модель фриланса, болезни и т. п.)	27,8	29,4
Нет, не замечаю такую тенденцию	33	28

Таблица 10

Все чаще проектные команды взаимодействуют по рабочим вопросам с помощью мобильных инструментов — мессенджеров, мобильных планировщиков, групповых чатов. Замечаете ли Вы данную тенденцию в знакомых проектных командах?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Да, часть рабочих коммуникаций проходит через мобильные каналы и инструменты	75,9	85,3
Нет, не замечаю такой тенденции	24,1	14,7

распределенных команд и с работой инженеров вне офиса компании. Однако не следует забывать, что в IT-отрасли граница между рабочими и личными делами сотрудников и без того довольно неопределенна. Удельный рост времени, посвящаемого работе с помощью смартфона, может иметь негативный оттенок с эмоционально-психологической точки зрения.

Современные тенденции в области обеспечения качества ПО и надежности его своевременной поставки и интеграции

Современные подходы к обеспечению качества ПО могут быть охарактеризованы следующим набором характеристик:

- наличие формализованной тестовой модели, охватывающей проверку соответствия функциональным и нефункциональным требованиям, процессы валидации и верификации каждого релиза;
- использование комплексной проверки качества каждого релиза (юнит-тестирования, автоматизированного тестирования, интеграционного тестирования);
- создание и использование различных предпромышленных сред (для различных типов тестирования, опытной эксплуатации и т. п.);
- наличие автоматических инструментов сборки, доставки и интеграции релизов ПО в такие среды.

Наиболее трудозатратной и вместе с этим эффективной практикой в обеспечении стабильно высокого качества ПО является создание автоматических тестов, которые способны проверять в релизе работоспособность основных автоматизируемых процессов. При этом наиболее распространенным уровнем в покрытии функционала автотестами является уровень 35...50 %, так как именно на данном пределе затраченные усилия наиболее эффективны в долгосрочной перспективе. В авторских исследованиях была изучена востребованность такого подхода в российских командах (табл. 11).

Опыт российских команд указывает на наличие существенных возможностей в повышении уровня

Таблица 11

Соответствует ли опыту Вашей команды / Вашему личному опыту значительный уровень (35...50 %) покрытия автоматическими тестами функционала системы?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Да, это соответствует моему (нашему) опыту	62	52,9
Нет, это не соответствует моему (нашему) опыту	36,7	45,6
Затрудняюсь ответить	1,3	1,5

конкурентоспособности в области качества ПО. Попрежнему значительное число команд не обладает достаточной уверенностью или проектными ресурсами для покрытия автоматическими тестами ключевого функционала.

Вместе с тем повышение качества ПО неотрывно связано с качеством кода и обслуживанием технического долга программного продукта. Наиболее сложной и при этом прогрессивной практикой уменьшения технического долга и улучшения качества кода является рефакторинг [14]. Современный подход к рефакторингу заключается в его выполнении на регулярной основе без привязки к датам крупных релизов. При этом работы по рефакторингу кода могут быть как составной частью задач в итерации (в RUP-образных методологиях), так и отдельными задачами (в составе спринта в "гибких" методологиях). Востребованность регулярного рефакторинга без привязки к датам крупных релизов в отечественной практике отображена в табл. 12.

Данные табл. 12 демонстрируют, что в опыте российских команд остается значительное пространство для совершенствования методов и средств управления качеством кода и техническим долгом с помощью рефакторинга кода, в том числе на регулярной основе. Наиболее точной рекомендацией в данном направлении может стать следующая установка.

При развитии продукта, имеющего как минимум четырехлетнюю историю коммерческой эксплуатации и функционального роста, необходимо:

- при использовании Agile-методологии посвящать 10...20 % рабочего времени в каждой итерации работе с техническим долгом и процессам рефакторинга;
- при использовании RUP-образной методологии выполнять в год 2—3 итерации по работе с техническим долгом, приостанавливая на время данных итераций функциональное развитие продукта.

Непрерывная поставка и интеграция (CI/CD) [15] как часть DevOps-подхода [3], отражает стремление занять доминирующее положение в мире, став наиболее востребованной практикой обеспечения высокого качества сборки и поставки программных продуктов. В основе концепции непрерывной поставки и интеграции лежит набор идей и технологий, обеспечивающих точную сборку, автоматизированное интеграционное тестирование, автоматическую поставку работающей версии в тестовую или промышленную среду для потребителей. При этом выполняется набор сопутствующих задач, в том числе управление качеством сборки и релиза, версионностью программных продуктов, минимизация сроков поставки новой версии ПО до потребителей. Так, сборочные конвейеры и интегрированные инструменты управления версионностью стали значительными конкурентными преимуществами в производстве ПО. Они обеспечивают существенную экономию ресурсов и повышают все ключевые показатели ПО — качество сборки финальных билдов, доступность инструментов тестирования и верификации, а также снижают время доставки новых версий до потребителей и т. п. Актуальность отмеченных выше подходов для российских команд отображена в табл. 13.

Данные в табл. 13 указывают на очевидную динамику роста востребованности подходов DevOps, включая непрерывную поставку и интеграцию новых релизов ПО. При этом практическая реализация процессов управления версиями ПО, сборки, автоматизированного тестирования, обновления и интеграции во многом зависит от используемых средств автоматизации. В последние годы наблюда-

Таблица 12

Используется ли в Вашем опыте такая практика, как проведение рефакторинга на регулярной основе "по расписанию"?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Да, мы используем рефакторинг как регулярную практику	27,9	19,1
Мы используем рефакторинг от случая к случаю	65,8	61,8
Мы не используем (почти не используем) рефакторинг кода	6,3	19,1

Таблица 13

Оцените востребованность DevOps-подхода и непрерывной поставки/интеграции (CI/CD) в разработке ПО среди знакомых Вам команд в регионе

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Подход DevOps (включая CI/CD) стал популярным и распространенным	57	76,5
Пока о подходах DevOps больше слов, чем реального применения	36,7	23,5
Данная тенденция не наблюдается	6,3	0

Таблица 14

Отмечаете ли Вы рост популярности инструмента Git в использовании командами Вашего региона?

Вариант ответа	Доля экспертов в группе, %	
	Исследование 1	Исследование 2
Согласен, отмечаю значительный рост популярности Git	89,9	91,2
Не согласен, не вижу изменений в его распространенности	8,9	7,3
Затрудняюсь ответить	1,2	1,5

ется лавинообразный рост популярности следующих программных инструментов:

- системы управления версиями Git;
- системы поддержки контейнеризации Docker.

При этом оба этих инструментальных средства отлично интегрированы между собой, что позволяет использовать их связку как исчерпывающее сочетание для обеспечения высокого качества управления релизами [16].

В авторских исследованиях была оценена популярность обоих инструментов среди российских команд. Так, инструментальное средство Git (*Distributed version control system*) продолжает демонстрировать высокий уровень популярности (табл. 14). Оно находится на пути получения наибольшей рыночной доли среди других инструментов управления версиями коммерческого ПО.

Инструментарий поддержки контейнеризации в процессах непрерывной поставки, обновления и интеграции — система Docker — также становится все более популярной в России (табл. 15). При этом его глобальная востребованность во многом обеспечена стратегическим альянсом его производителя с гигантом отрасли — корпорацией IBM. Некоторые исследователи предсказывают данному инструментарию глобальное превосходство в ближайшем будущем. Причина в его ключевых свойствах — универсальности, легкости в разворачивании и эксплуатации, готовой интеграции с другими инструментами обеспечения современного управления поставками релизов [17].

Таблица 15

Отмечаете ли Вы в своей команде (и командах вокруг) рост популярности подходов и инструментов контейнеризации?

Вариант ответа	Доля экспертов в группе, % (исследование 2)
Да, все больше команд использует именно Docker	67,6
Контейнеризация становится популярнее, но используются другие инструменты	20,6
Не отмечаю рост популярности ни инструмента, ни подхода	10,3
Затрудняюсь ответить	1,5
Примечание: в исследовании 2017 г. этот вопрос перед экспертами не ставился.	

Безусловно, рынок инструментов DevOps будет переживать значительные изменения, так как крупные корпорации и сообщества разработчиков будут предлагать новые решения. Уже сейчас у инструментов Git и Docker есть серьезные нишевые конкуренты. Однако в настоящее время российский опыт, установленный в авторских исследованиях, в целом соответствует мировым тенденциям быстрого роста популярности использования данных программных продуктов.

Заключение

Российская экономика с ее отчетливым экспортным уклоном обладает довольно скромными финансовыми результатами в области экспорта продуктов и услуг IT-отрасли. Наряду с Китаем и Индией российский IT-рынок ПО является скорее источником экспорта специалистов, а не лучших в мире технологий и программных решений. Вместе с тем глобальная конкурентная среда, несмотря ни на какие протекционистские меры, заставляет IT-компании улучшать все производственные показатели, в том числе на внутреннем рынке.

Приведенные результаты авторских исследований 2017 и 2019 гг. позволяют прийти к перечисленным далее выводам.

1. Переход к "гибким" и "гибридным" методологиям в производстве ПО во многом завершен. Отечественные IT-компании и IT-подразделения внутренней разработки обладают достаточным опытом и знаниями о целесообразности использования тех или иных артефактов и процессов из "легковесных" методологий, способны построить/построили современные производственные модели.

2. Уровень формализации проектного управления в разработке ПО находится в положительной динамике. При этом он по-прежнему обладает существенным запасом для дальнейшего роста, что должно помочь в увеличении глобальной конкурентоспособности отечественных IT-компаний.

3. Текущие подходы отечественных команд в области современной организации командной работы и использования инструментов рабочих коммуникаций находятся в процессе трансформации, что соответствует мировой практике.

4. Использование современных ресурсоемких методик обеспечения постоянно высокого качества ПО

находится в отрицательной динамике. Возможным объяснением такого явления может быть экономия проектных средств в IT-компаниях, работающих в условиях длительной экономической стагнации.

5. Современные практики непрерывной интеграции и обновления, а также реализующие их инструментальные средства становятся все более востребованными в российских командах разработчиков.

Следует отметить, что полученные результаты и выводы характеризуют отечественную программную инженерию, как соответствующую развивающимся и догоняющим рынкам, очень восприимчивую и открытую к лучшим мировым практикам. Следует также предположить, что ориентация отечественных команд разработки на условно-бесплатные инструментальные средства автоматизации производственных процессов с открытым исходным кодом будет увеличиваться.

Однако отметим, что российские IT-компании находятся в режиме существенной экономии средств при долгосрочных производственных инновациях. Это обстоятельство отражается в удешевлении методов обеспечения качества ПО и продолжающемся отсутствии организационно оформленных практик проектного управления и мониторинга в каждой IT-компании. При этом общий технологический уровень команд, в том числе на уровне инструментального оснащения, соответствует ожиданиям глобальных корпоративных потребителей, что оставляет существенные шансы на увеличение экспортной выручки отрасли.

Список литературы

1. **Вроблевски Л.** Сначала мобильные! М.: Манн, Иванов и Фербер (МИФ), 2012. 160 с.
2. **Кент Б.** Экстремальное программирование: разработка через тестирование = Test-driven Development. СПб.: Питер, 2003. 294 с.
3. **Дэвис Дж., Дэниелс К.** Философия DevOps. Искусство управления IT. СПб.: Питер, 2017. 610 с.
4. **Комаров Н. М., Пащенко Д. С.** Современная высокотехнологичная компания в IT-отрасли: краткий обзор // Вестник Евразийской науки. 2019. Т. 11, № 4. URL: https://esj.today/PDF/58SAVN419.pdf
5. **Pashchenko D. S.** Russian local trends in software development and implementation // Мир новой экономики. 2017. № 3. С. 29—35.
6. **Agile-**инструменты для команд разработчиков. URL: https://www.atlassian.com/ru/software/jira/agile
7. **Щукова К. Б.** Методология IBM Rational Unfied Process как инструмент для создания эффективных систем // Современная техника и технологии. 2015. № 11. URL: http://technology.snauka.ru/2015/11/8172
8. **Ableson F., Sen R., King C.** Android in Action, Second Edition (2nd ed.). Manning Publications, 2011. 575 p.
9. **Пащенко Д. С.** Отражение в российской практике мировых тенденций в технологиях, средствах и подходах в разработке программного обеспечения // Программная инженерия. 2017. Т. 8, № 8. С. 339—344.
10. **Брагина Т. И., Табунщик Г. В.** Сравнительный анализ итеративных моделей разработки программного обеспечения // Радіоелектроніка, інформатика, управління. 2010. № 2 (23). С. 130—138.
11. **Поппендик М., Поппендик Т.** Бережливое производство программного обеспечения. От идеи до прибыли. М.: Вильямс, 2010. 256 с.
12. **Богданов В. В.** Управление проектами. Корпоративная система — шаг за шагом. М.: Манн, Иванов и Фербер, 2012. 248 с.
13. **Пащенко Д. С.** Географически распределенные команды: естественные и организационные особенности проектов разработки программного обеспечения // Программная инженерия. 2017. Т. 8, № 2. С. 88—95.
14. **Fowler M.** Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2018. 448 p.
15. **Humble J., Farley D.** Continuous Delivery: reliable software releases through build, test, and deployment automation. Addison-Wesley Professional, 2011. 512 p.
16. **Мил И., Сейрс Э.** Docker на практике. М.: ДМК Пресс, 2020. 516 с.
17. **Suleman A.** Docker And Kubernetes: Furthering The Goals Of DevOps Automation // Forbes. Innovations. 10-10-2018. URL: https://www.forbes.com/sites/forbestechcouncil/2018/10/10/docker-and-kubernetes-furthering-the-goals-of-devops-automation

Modern World-Wide Practices in Software Engineering and Software Testing in the Russian IT Industry: Results of Researches (2017 and 2019)

D. S. Pashchenko, denpas@rambler.ru, Moscow, 125368, Russian Federation

Corresponding author:

Pashchenko Denis S., Moscow, 125368, Russian Federation
E-mail: denpas@rambler.ru

Received on December 17, 2019
Accepted on February 10, 2020

The development of software engineering is a very fast process. Already long-growing trends in the transition to "flexible" and "lightweight" software development paradigms are joined by new technologies of version control management and methods of continuous integration and delivery of software releases. New forms of organizing development teams, including geographically distributed teams, work outside of offices, and mobile planning and communications tools, are gaining significant importance. This article summarizes some results of two author studies of 2017 and 2019 about the reflection of modern world practices in software engineering and software quality

management in Russian IT companies. About 150 engineers, analysts, project managers from all federal districts of the Russian Federation took part in the researches. The article presents the corresponding analysis and conclusions about the dynamics of the main processes and the similarities of technological and organizational shifts in Russian software engineering in comparison with global trends.

Keywords: *software engineering, software quality management, organization of software teams, software project management*

For citation:

Pashchenko D. S. Modern World-Wide Practices in Software Engineering and Software Testing in the Russian IT Industry: Results of Researches (2017 and 2019), *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 67—76

DOI: 10.17587/prin.11.67-76

References

1. **Vroblevski L.** *Mobile First,* Moscow, Mann, Ivanov i Ferber (MIF), 2012, 160 p. (in Russian).
2. **Kent B.** *Extreme Programming: Test Driven Development*, Saint-Peterburg, Piter, 2013, 294 p. (in Russian).
3. **Devis J., Deniels K**. *DevOps Philosophy. Art of IT management*, Saint-Peterburg, Piter, 2017, 610 p. (in Russian).
4. **Komarov N. M., Pashchenko D. S.** Modern high-tech IT company, *Vestnik Evrazijskoj nauki,* 2019, vol. 11, no. 4, available at: https://esj.today/PDF/58SAVN419.pdf (in Russian).
5. **Pashchenko D. S**. Russian local trends in software development and implementation, *Mir novoj ekonomiki,* 2017, no. 3, pp. 29—35 (in Russian).
6. **Agile**-instrument for software development teams, available at: https://www.atlassian.com/ru/software/jira/agile
7. **Shchukova K. B.** IBM Rational Unified Process Methodology for software development, *Sovremennaya tekhnika i tekhnologii*, 2015, no. 11, available at: http://technology.snauka.ru/2015/11/8172 (in Russian).
8. **Ableson F., Sen R., King C.** *Android in Action,* Second Edition (2nd ed.), Manning Publications, 2011, 575 p.
9. **Pashchenko D. S.** Reflection in the Russian Practice of World Trends in Technologies, Tools and Approaches to Software Development, *Programmnaya Ingeneria,* 2017, vol. 8, no. 8, pp. 339—344 (in Russian).
10. **Bragina T. I., Tabunshchik G. V.** Comparing analysis of interactive software development process models, *Radioelektronika, informatika, upravlinnya*, 2010, no. 2 (23), pp. 130—138 (in Russian).
11. **Poppendik M., Poppendik T.** *Lean Software Development: from idea to profit*, Moscow, Williams, 2010, 256 p. (in Russian).
12. **Bogdanov V. V.** *Project management. Corporate system*, Moscow, Mann, Ivanov i Ferber, 2012. 248 p. (in Russian).
13. **Pashchenko D. S.** Research in CEE-region: Changes Implementation in Software Production, *Programmnaya Ingeneria,* 2017, vol. 8, no. 2, pp. 88—95 (in Russian).
14. **Fowler M.** *Refactoring: Improving the Design of Existing Code,* Addison-Wesley, 2018, 448 p.
15. **Humble J., Farley D.** *Continuous Delivery: reliable software releases through build, test, and deployment automation*, Addison-Wesley Professional, 2011, 512 p.
16. **Miell I., Sayers A. H.** Docker in Practice, 2020, 516 p. (in Russian).
17. **Suleman A.** Docker And Kubernetes: Furthering The Goals Of DevOps Automation, *Forbes. Innovations*, 10-10-2018, available at: https://www.forbes.com/sites/forbestechcouncil/2018/10/10/docker-and-kubernetes-furthering-the-goals-of-devops-automation

ИНФОРМАЦИЯ

УДК 681.5; 004.5 DOI: 10.17587/prin.11.77-85

М. В. Бобырь, д-р техн. наук, проф. кафедры, fregat_mn@rambler.ru,
А. А. Дородных, аспирант, alex.dorodnych@mail.ru,
А. С. Якушев, аспирант, alekseyakushev@yandex.ru,
В. А. Булатников, аспирант, rb465687@gmail.com, Юго-Западный государственный университет, г. Курск

Аппаратно-программный мехатронный комплекс для фиксации подвижных объектов

Представлена аппаратно-программная модель мехатронного комплекса для фиксации подвижных объектов. Показана циклограмма, описывающая ход пневматических цилиндров за определенное время. Описан подход к созданию человеко-машинного интерфейса для визуализации процесса перемещения исполнительных механизмов на экране интерактивной сенсорной панели. Рассмотрены математическая модель и программный код управления мехатронным комплексом в автоматическом режиме. Приведен сравнительный анализ автоматического и ручного режимов управления, и на основе регрессионного анализа дана оценка прогнозирования момента времени, когда совмещаются оси пневматического схвата и подвижного объекта.

Ключевые слова: *мехатронный комплекс, программный код, программируемый логический контроллер, метки-идентификаторы, человеко-машинный интерфейс, пневматические исполнительные механизмы*

Введение

В статье представлена аппаратно-программная модель мехатронного комплекса (МК), предназначенного для автоматизации технологических процессов и выполняющего следующие технологические операции (ТО): перемещение исполнительных механизмов (ИМ), обнаружение, фиксация и складирование подвижных объектов. В раннее опубликованном научном исследовании [1] авторским коллективом были рассмотрены вопросы синергетического объединения компонентов МК в единую систему и процесс его функционирования в ручном режиме управления. Ручной режим управления МК подразумевает, что человек-оператор осуществляет управление перечисленными выше ТО с помощью команд на экране интерактивной сенсорной панели (ИСП). Автоматический режим управления отличается тем, что человек-оператор запускает МК с помощью реальной и/или виртуальной кнопок, а остальные действия происходят за счет заложенной в программируемый логический контроллер (ПЛК) программы. При выполнении рабочего цикла в данном режиме функционирования МК наиболее сложной технологической операцией является фиксация подвижного объекта. Успешное выполнение этой операции зависит от прогнозирования и контроля траектории перемещения пневматических ИМ, имеющих нелинейные характеристики [2]. Если в результате перемещений центральные оси подвижного объекта и пневматического схвата не совмещены, то объект с высокой степенью вероятности не зафиксируется в губках схвата. При этом нарушается дальнейший ход выполнения ТО, что влияет на работоспособность МК [3—6]. В ручном режиме человек-оператор с помощью визуального контроля определяет момент совмещения осей пневматического схвата относительно подвижного объекта, затем выполняет команду на захват подвижного объекта. В автоматическом режиме управления МК для фиксации подвижного объекта используется оптический датчик. Он фиксирует только наличие подвижного объекта в зоне обнаружения, а сам момент совмещения осей контролируемых объектов он определить не может. Таким образом, для автоматического режима управления МК возникает проблемная ситуация, связанная с нахождением баланса между минимальным временем, необходимым для совмещения осей подвижного объекта и пневматического схвата, и временем задержки для реализации команды на фиксацию подвижного объекта, регулируемым с помощью системных таймеров.

Исследование, результаты выполнения которого представлены в настоящей работе, было посвящено решению следующих задач: разработка программного кода для управления МК в автоматическом режиме [7]; создание модели человеко-машинного интерфейса и верификация ее работоспособности; анализ кинематической модели; сравнение полученных экспериментальных результатов для двух режимов управления; оценка прогнозирования момента времени, когда совмещаются оси пневматического схвата и подвижного объекта.

Состав и принцип действия мехатронного комплекса

К основным компонентам МК относятся: пневматические исполнительные механизмы, управляемые с помощью программируемого логического контроллера Siemens S7-1200; система обнаружения подвижных объектов и контроля перемещения ИМ в режиме реального времени, состоящая из магнитных датчиков положения (ДП) и оптического датчика (ОД); интерактивная сенсорная панель Simatic HMI KTP 400 basic; Ethernet-коммутатор Scalance XB005; персональная электронно-вычислительная машина (ПЭВМ); пневмоостров, фильтр-регулятор давления, манометры и компрессор. На рис. 1 изображены внешний вид МК и состав его компонентов (ПЭВМ и компрессор не указаны) [8—10]. Принцип работы МК, состоящий из последовательных перемещений, выполняемых ИМ в автоматическом режиме функционирования в зависимости от сигнальных состояний ДП и ОП в сочетании с временем задержки, представлен в виде циклограммы (рис. 2).

Для осуществления технологической операции фиксации в автоматическом режиме функционирования МК необходимо выполнить перемещение исполнительных механизмов со второго по пятый такт (см. рис. 2). После удара по подвижному объекту штоком цилиндра одностороннего действия шарик

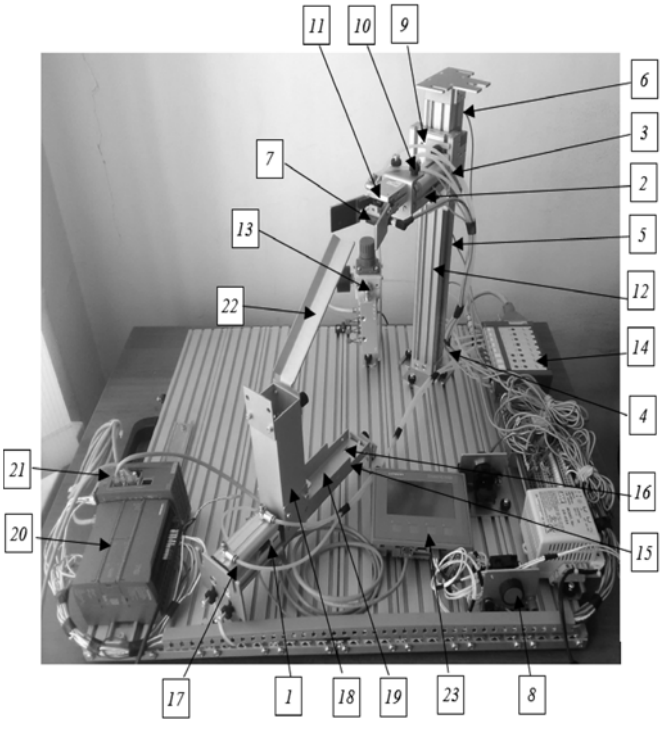

Рис. 1. Внешний вид и состав МК:

1—6 — магнитные датчики положения; *7* — оптический датчик; *8* — кнопка "Старт"; *9* — бесштоковый цилиндр с кареткой; *10* — цилиндр двустороннего действия; *11* — пневматический схват; *12* — направляющая бесштокового цилиндра; *13* — фильтр-регулятор; *14* — пневмоостров; *15* — дроссель-регулятор; *16* — сопло; *17* — цилиндр одностороннего действия; *18* — лоток хранения; *19* — направляющая; *20* — программируемый логический контроллер Siemens S7-1200; *21* — Ethernet-коммутатор Scalance XB005; *22* — желоб; *23* — интерактивная сенсорная панель

перемещается из лотка хранения по направляющей в сопло и начинает парить в потоке сжатого воздуха (с третьего по пятый такт). Каретка бесштокового цилиндра (БЦ) с цилиндром двустороннего действия (ЦДД) и пневматическим схватом (ПС) перемещаются до момента обнаружения парящего (подвижного) объекта оптическим датчиком. Когда ОД обнаружил объект и совпали центры координат объекта и пневматического схвата (рис. 3), выполнено условие для совмещения осей. В программном коде, загруженном в программируемый логический контроллер из ПЭВМ через Ethernet-коммутатор, с помощью системных таймеров формируется необходимая задержка по времени. Затем объект с гарантированной надежностью фиксируется с помощью губок пневматического схвата. После этого подача воздуха в сопло прекращается и зафиксированный объект перемещается для складирования в лоток хранения (с шестого по десятый такт). Исполнительные механизмы возвращаются в стартовую позицию (см. рис. 1) перед началом нового рабочего цикла.

При проектировании системы управления МК авторским коллективом была разработана модель человеко-машинного интерфейса (ЧМИ) в пакете специализированного программного обеспечения Simatic WinCC, входящего в интегрированную среду для систем автоматизации технологических процессов Totally Integrated Automation (TIA) Portal V14.1. Визуализация процесса перемещения исполнительных механизмов, обнаружения, фиксации и складирования подвижного объекта реализована на экране интерактивной сенсорной панели. На рис. 4 дан вид в рабочем окне оператора — кнопки управления и фактическое местоположение исполнительных механизмов МК. Человек-оператор контролирует процесс перемещения каретки бесштокового цилиндра и цилиндра двустороннего действия с пневматическим схватом. На экране прямоугольниками изображаются ДП и ОД с функцией цветовой индикации, указывающей на текущее местоположение ИМ относительно подвижного объекта.

Если ДП находится в значении логической 1, т. е. срабатывание, а ИМ — в контактной области данного датчика, то соответствующий датчику прямоугольник загорается зеленым цветом. В случае, когда ИМ отсутствует в контактной области и контакт датчика разомкнут, т. е. находится в значении логического 0, индикация меняется на красный цвет [11—13].

Кинематическая модель мехатронного комплекса

Кинематическая модель разработана для описания перемещений исполнительных механизмов МК. Она позволяет прогнозировать время и координаты для совмещения осей пневматического схвата и подвижного объекта. Ускорение при перемещении пневматического схвата вычисляется по формуле

$$a_{\text{тек}} = \left(A_0 + \sin \frac{v\pi}{v_{\max}} \right) \times a_{\max}, \qquad (1)$$

где $a_{\max}$ — максимальное значение ускорения ИМ, м/с^2; v — текущая скорость перемещения ИМ, м/с; $v_{\max}$ — максимальная скорость перемещения ИМ, м/с, A_0 — регулировочный коэффициент.

Механизм		Состояние и положение механизмов	Такт									
			1	2	3	4	5	6	7	8	9	10
Кнопка	Старт	вкл. выкл.										
	Стоп	вкл. выкл.										
Бесштоковый цилиндр (положение)		верхнее среднее нижнее										
Бесштоковый цилиндр (направление движения)		вверх вниз отсутствует										
Пневм. схват		открыт закрыт										
Цилиндр двустороннего действия		выдвинут втянут										
Сопло (подача воздуха)		вкл. выкл.										
Цилиндр одностороннего действия		выдвинут втянут										
Оптический датчик		объект обнаружен объект не обнаружен										
Магнитные датчики положения	ДП 1		●	●	○	●	●	●	●	●	●	●
	ДП 2		●	●	○	○	○	●	●	○	○	●
	ДП 3		○	○	●	●	●	○	○	●	●	○
	ДП 4		●	●	●	●	●	●	●	●	●	●
	ДП 5		●	●	○	●	●	●	●	●	●	●
	ДП 6		○	●	●	●	●	●	○	●	●	●
Время выполнения цикла, с			0,87	2,04	0,84	1,35	1,03	1,18	3,81	0,83	0,94	1,01
Примечание		начальное состояние – – – – –, сработка датчиков положения герконов – ○, нет сработки герконов – ●										

Рис. 2. Циклограмма последовательности перемещений исполнительных механизмов МК

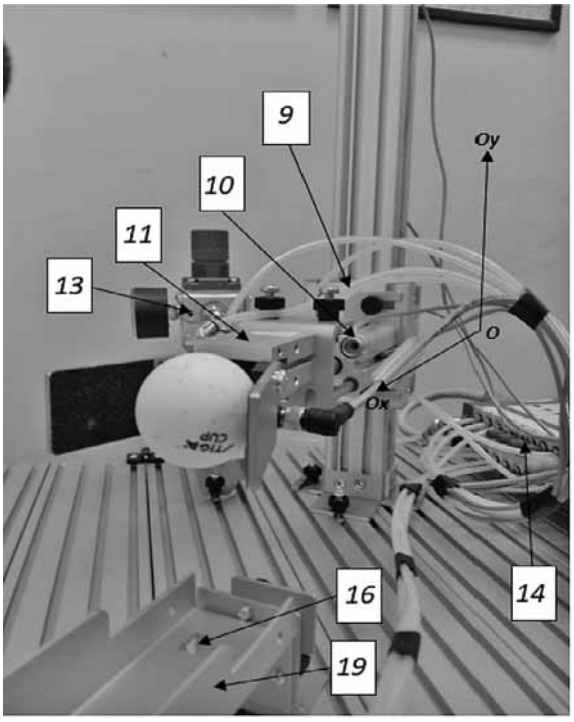

Рис. 3. Совмещение центральных осей подвижного объекта и ПС

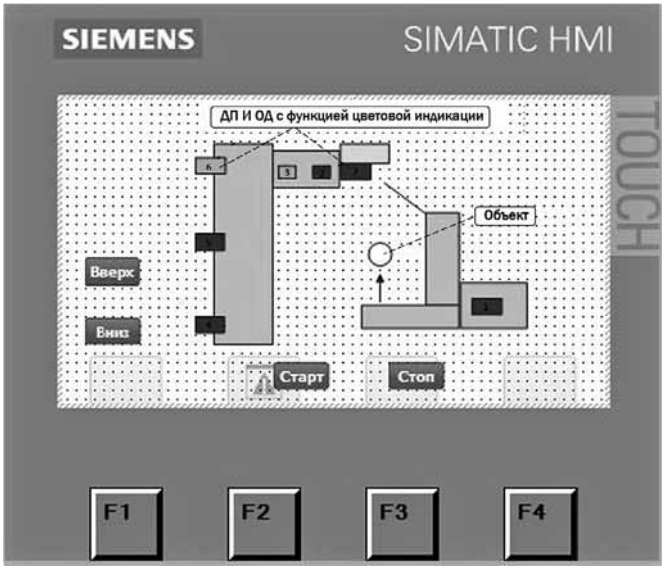

Рис. 4. Вид в рабочем окне оператора

Скорость перемещения ИМ в заданный период времени описывается уравнением

$$V_{\text{тек}} = v_{\text{пред}} + \frac{a_{\text{тек}}}{t}, \qquad (2)$$

где $v_{пред}$ — скорость на предыдущем шаге интегрирования, м/с; t — время перемещения исполнительных механизмов, с.

Текущее положение пневматического схвата рассчитывается для координат *Sx* и *Sy*, оно равно расстоянию перемещения ИМ вдоль осей *Ox*, *Oy* и в общем виде представлено как

$$S = S_{пред} + \frac{v_{тек}}{t}, \quad (3)$$

где $S_{пред}$ — предыдущее положение ПС, м.

Следующие две формулы позволяют определять координаты, в которых находится подвижный объект в текущий момент времени:

$$x = x_0 + \sin\left(x_{пред} + \frac{v_x}{t}\right) \times l_x, \quad (4)$$

$$y = y_0 + \sin\left(y_{пред} + \frac{v_y}{t}\right) \times l_y, \quad (5)$$

где x_0, y_0 — положение подвижного объекта при совмещении осей, м; $x_{пред}$, $y_{пред}$ — предыдущие положения в момент колебаний, м; l_x, l_y — размах колебаний, м.

Исходя из формул (3)—(5), пневматический схват фиксирует подвижный объект при выполнении условия

$$\left(|S_x - x_0| < a_x\right) \& \left(|S_y - y_0| < a_y\right), \quad (6)$$

где a_x, a_y — расстояние, м, по осям *Ox*, *Oy* между центром ПС и центром подвижного объекта.

Программный код для управления мехатронным комплексом в автоматическом режиме

Для автоматизации процессов функционирования МК авторским коллективом был разработан программный код управления МК, обеспечивающий синхронизацию выполняемых ТО в соответствии с кинематической моделью и циклограммой (см. рис. 2). Для этого в среде Simatic Step 7 (TIA Portal v14) созданы метки-идентификаторы процесса (табл. 1), связанные с адресами в памяти и каналами ввода/вывода ПЛК. На рис. 5 в окне инспектора Simatic Step 7 на вкладке устройства показан пример конфигурации ПЛК с назначенными метками-идентификаторами. Входные переменные I относятся к каналам ввода информации в ПЛК и отвечают за обработку данных, поступающих от оптического датчика и датчиков положения. Выходные переменные Q связаны с каналами вывода информации из ПЛК и используются для передачи управляющего воздействия на исполнительные механизмы МК. Локальные переменные M используются для связи функциональных кнопок на экране ИСП оператора с ПЛК (см. рис. 4).

В табл. 1 представлены метки-идентификаторы, необходимые для стабилизации осей пневматического схвата и подвижного объекта при выполнении технологической операции фиксации подвижного объекта. В столбцах даны: символьный адрес; обозначение (название); тип данных; примечание, в котором можно указывать комментарии к метке-идентификатору. Например, если оптический датчик

Таблица 1

Метки-идентификаторы процесса

Переменные	Символьный адрес	Обозначение	Тип	Примечание
Входные	%I0.2	arm_outside	bool	Пневматический схват выдвинут
	%I0.3	arm_inside	bool	Пневматический схват втянут
	%I0.5	laz_datchik	bool	Оптический датчик
	%I0.6	gerkon_vpered	bool	Магнитные датчики положения (геркон)
	%I0.7	gerkon_up	bool	
	%I1.0	gerkon_niz	bool	
Выходные	%Q0.1	shvat	bool	Пневматический схват
	%Q0.0	arm	bool	Выдвижение ПС
	%Q0.5	down	bool	Перемещение ПС вниз
Локальные	%M0.1	arm_down	bool	Перемещение (ИСП)
	%M10.2	trig2	bool	Отмена фиксации
	%M10.4	trig4	bool	Сброс перемещения
	%M5.2	c4_1	bool	Перемещение с выдвижением ПС
	%DB2	timer_2	real	Системный таймер задержки
	%DB6	timer_7	real	

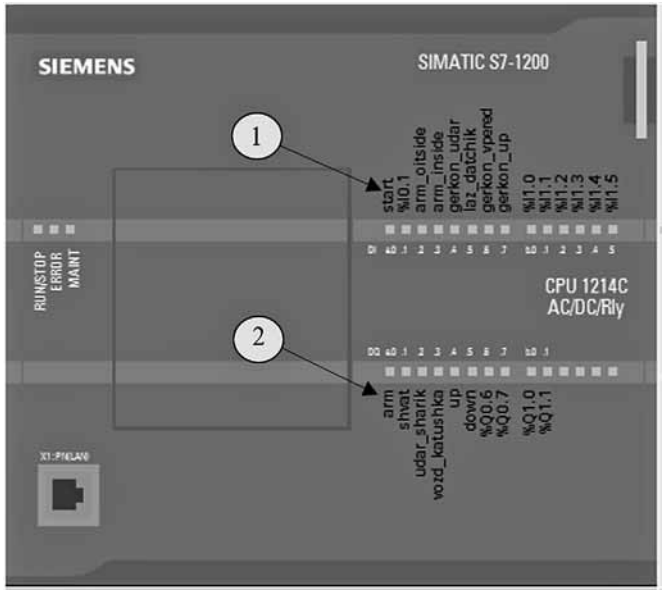

Рис. 5. Пример конфигурации ПЛК с назначенными метками-идентификаторами:

1 — входные переменные; *2* — выходные переменные

подключен к пятому входу ПЛК (см. рис. 5), то он будет обозначен переменной вида %I0.5, где символ I указывает, что это входной сигнал, 0.5 — что датчик подключен к пятому входу ПЛК. Символ % указывает, что переменная I0.5 является абсолютным операндом, которому присваивается имя, например, laz_datchik.

При написании программного кода использован язык релейно-контактных схем (LD — *Ladder Diagram*). Программный код управления МК состоит из сегментов, которые соединяются в логические цепочки с использованием графических элементов языка (контактов, катушек, таймеров, счетчиков и т. д.). При этом последовательно соединенные контакты выполняют логическую операцию "И", а параллельно соединенные контакты — логическую операцию "ИЛИ". На рис. 6 представлена реализация технологической операции фиксации подвижного объекта. Для этого необходимо управлять временем срабатывания пневматического схвата, которое задано в циклограмме (см. рис. 2) на четвертом такте и равняется 1,35 с. Данное время прогнозируется с помощью кинематической модели. В программном коде данное значение задается суммированием задержек на системных таймерах %DB6 "timer_7" и %DB2 "timer_2". Общее время задержки составляет 1 + 0,35 = 1,35 с (см. рис. 6).

При этом таймер %DB6 "timer_7" через 0,35 с устанавливает катушку сброса %Q0.5 в активное состояние. Каретка бесштокового цилиндра с цилиндром двустороннего действия и пневматическим схватом прекращают движение, оси пневматического схвата и подвижного объекта центрируются относительно друг друга (см. рис. 3). В результате шарик фиксируется губками пневматического схвата.

Экспериментальные исследования

Кинематическая модель МК была разработана с помощью программного обеспечения MATLAB в компоненте Simulink. В процессе ее реализации были проведены два эксперимента. Во время первого эксперимента с использованием выражений (1)—(6) строились трехмерная и двухмерная модели для анализа момента совмещения осей ПС и подвижного объекта (рис. 7). Данный эксперимент позволяет определить местоположение ПС, в котором происходит фиксация подвижного объекта, а также спрогнозировать время срабатывания пневматического схвата (см. рис. 7, *б*), которое задается в программном коде с помощью системных таймеров (см. рис. 6). Последнее значение учитывается в системных таймерах %DB6 "timer_7" и %DB2 "timer_2" (см. рис. 6) и указывается в циклограмме на четвертом такте (см. рис. 2, t = 1,35 с). Точность прогноза времени фиксации подвижного объекта с помощью разработанной кинематической модели была оценена на основе регрессионного метода и коэффициентов MSE, MAPE, R^2 [14]. Для этого были синтезированы уравнения линейной и степенной регрессии (рис. 8, см. вторую сторону обложки) [11, 14], показывающие взаимосвязь между временем и перемещением исполнительных механизмов $t = f(S)$.

На рис. 8, *а* (см. вторую сторону обложки) представлен анализ перемещения во времени каретки бесштокового цилиндра вдоль оси Y (см. рис. 7, *а*, позиция *1*). Время перемещения ИМ (см. рис. 8, *а* — ось ординат) составляет 2,04 с и указывается в циклограмме на втором такте (см. рис. 2). На рис. 8, *б* представлен анализ перемещения во времени цилиндра двухстороннего действия вдоль оси X (см. рис. 7, *а*, позиция *2*). Время перемещения ИМ (см. рис. 8, *б* — ось ординат) составляет 0,84 с и указывается в циклограмме на третьем такте (см. рис. 2). На рис. 8, *в* представлен анализ перемещения каретки бесштокового цилиндра вдоль оси Y (см. рис. 7, *а*, позиция *3*) до момента совмещения осей подвижного объекта и пневматического схвата. Время перемещения ИМ (см. рис. 8, *в* — ось ординат) составляет 1,35 с и указывается в циклограмме на четвертом такте (см. рис. 2).

На рис. 7, *б* по оси абсцисс выделено время перемещения исполнительных механизмов (см. рис. 2, такты со второго по четвертый) для осуществления технологической операции фиксации подвижного объекта. Указанный промежуток времени составил 4,23 с, он спрогнозирован с помощью разработанной кинематической модели.

В табл. 2 представлен анализ показателей прогнозирования момента времени, когда совмещаются оси пневматического схвата и подвижного объекта (см. рис. 3).

На основе данных табл. 2 можно сделать вывод, что предложенная кинематическая модель по отношению к регрессионным моделям позволяет точнее прогнозировать время совмещения осей для фиксации подвижного объекта в пневматическом схвате. Другой положительной особенностью кинематической модели является ее неразрывность. При использовании регрессионных моделей в точках изменения

осей перемещения исполнительных механизмов МК происходит разрыв точек траектории. Как следствие, если строить регрессионные модели для всего процесса перемещений исполнительных механизмов, то статистические коэффициенты будут иметь более низкие значения.

Во втором эксперименте в течение 100 рабочих циклов были проконтролированы: строгое соблюдение ТО в соответствии с циклограммой (см. рис. 2); своевременное включение различных индикаций; правильность срабатывания датчиков; корректность выставленных задержек времени для данных дат-

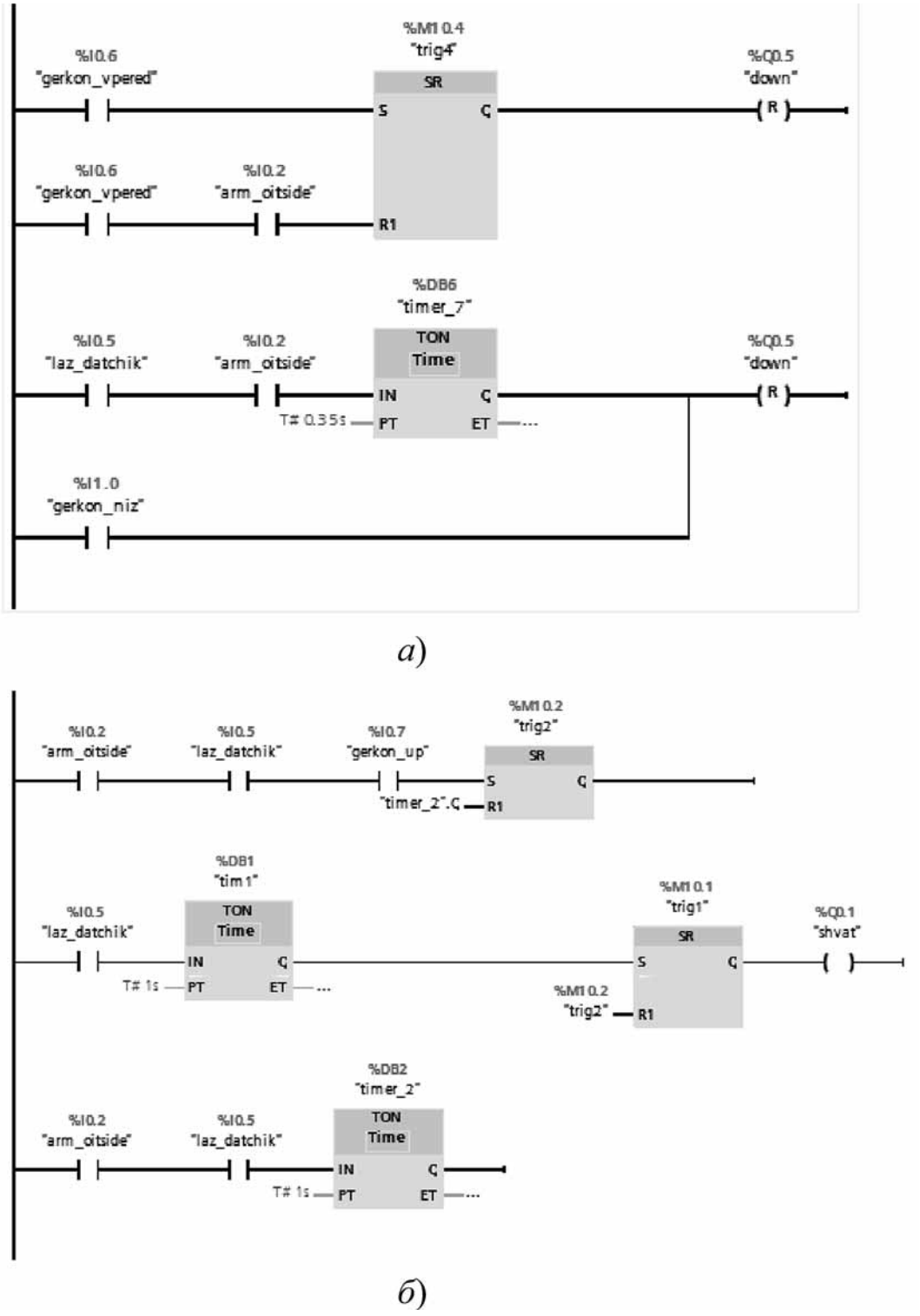

Рис. 6. Сегменты программного кода для реализации технологической операции фиксации:
а — стабилизация осей ИМ; *б* — захват подвижного объекта

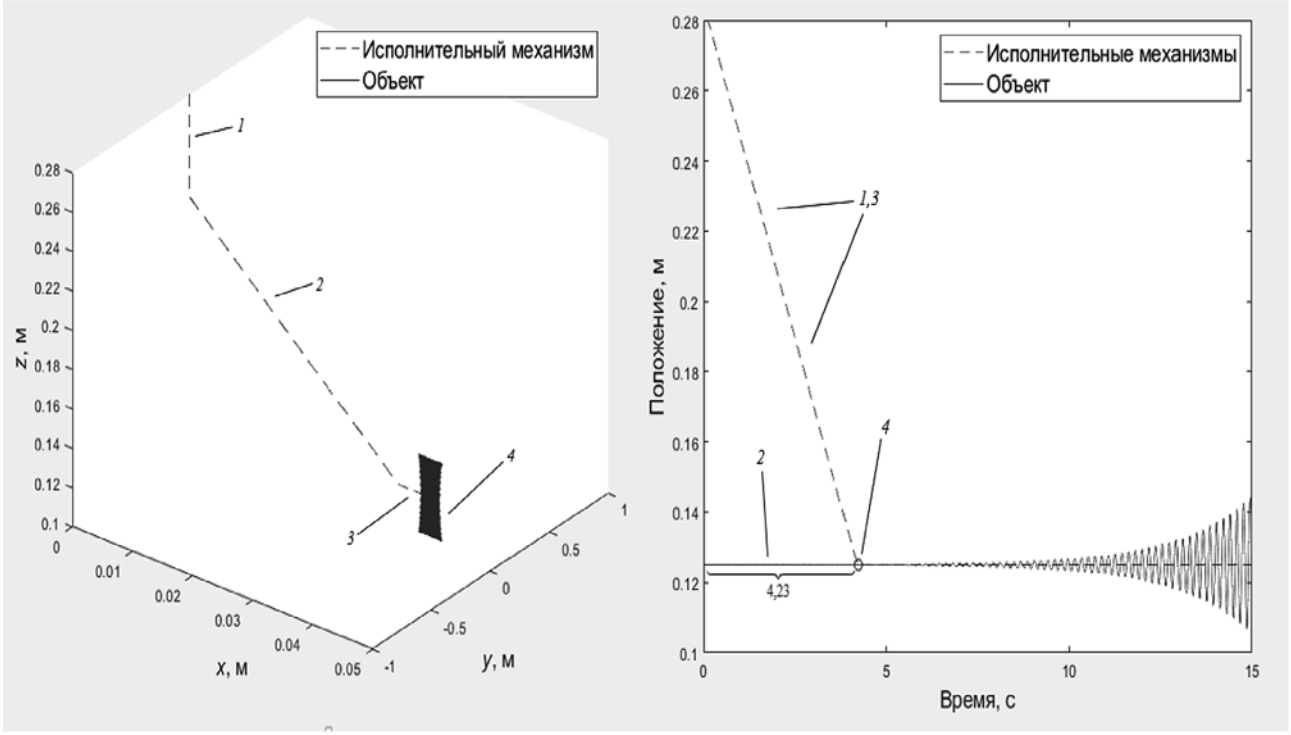

Рис. 7. Кинематическая модель перемещения пневматического схвата и подвижного объекта:
а — 3D-модель; *б* — 2D-модель; *1, 3* — перемещение каретки бесштокового цилиндра по оси *Oy*; *2* — перемещение каретки бесштокового цилиндра по оси *Ox*; *4* — момент совмещения осей

чиков. По результатам эксперимента выявлено, что при выполнении МК 100 циклов во всех случаях правильно выполнена последовательность технологических операций. Этот факт свидетельствует о работоспособности программного кода, правильности подключения и отладки компонентов МК [15—19]. Для сравнения времени выполнения ТО при управлении МК в ручном и автоматическом режимах проанализируем табл. 3, составленную на основании выборки из десяти полученных результатов.

Эксперимент в ручном режиме был проведен в соответствии с циклограммой (см. рис. 2). Зафиксировано время выполнения цикла в секундах с учетом реакции оператора. Среднее арифметическое измеряемой величины равно 21,94 с. Измерение времени выполнения ТО в автоматическом режиме показало следующий результат: среднее арифметическое измеряемой величины в автоматическом режиме составило 13,97 с; по сравнению с ручным режимом получено ускорение процесса выполнения МК рабочего цикла на 7,97 с.

Таблица 2

Регрессионный анализ

Метод анализа	Перемещения ИМ	MSE, с	MAPE	R^2	Точность, %
Линейная регрессия	Каретка БЦ (рис. 8, *а*, см. вторую сторону обложки)	0,0002	0,56	0,9995	99,44
	Пневматический схват (рис. 8, *б*, см. вторую сторону обложки)	0,0005	1,92	0,9978	98,08
	Каретка БЦ (рис. 8, *в*, см. вторую сторону обложки)	0,0007	6,55	0,9908	93,45
Степенная регрессия	Каретка БЦ (рис. 8, *а*, см. вторую сторону обложки)	0,039	5,11	0,974	94,89
	Пневматический схват (см. рис. 8, *б*, см. вторую сторону обложки)	0,003	1,97	0,9989	98,03
	Каретка БЦ (рис. 8, *в*, см. вторую сторону обложки)	0,013	6,03	0,9857	93,97
Кинематическая модель	Каретка БЦ (рис. 8, *а*, см. вторую сторону обложки)	0,0002	0,37	0,9996	99,63
	Пневматический схват (см. рис. 8, *б*, см. вторую сторону обложки)	0,0004	1,99	0,999	98,01
	Каретка БЦ (рис. 8, *в*, см. вторую сторону обложки)	0,0002	5,63	0,991	96,37

Таблица 3

Сравнение времени выполнения ТО при управлении МК

Режим управления	Номер цикла										Среднее арифметическое значение, с
	1	2	3	4	5	6	7	8	9	10	
	Время выполнения операции, с										
Ручной	22,00	22,04	21,97	22,00	21,95	21,83	21,86	21,90	21,89	21,97	21,94
Автоматический	14,00	13,96	14,04	13,92	14,02	14,00	13,92	13,94	13,90	14,02	13,97

Заключение

Рассмотрены аппаратно-программные элементы МК, выполняющего технологические операции в двух режимах управления — в ручном и автоматическом. Представлена циклограмма, отображающая последовательность перемещений исполнительных механизмов и датчиков МК. Создана модель человеко-машинного интерфейса и ее программная реализация, предоставляющая оператору информацию о текущем местоположении исполнительных механизмов, сигнальных состояниях датчиков с помощью цветовой индикации и возможностью управления МК на экране ИСП. Разработан программный код на языке релейно-контактных схем, обеспечивающий выполнение перемещений исполнительных механизмов в целях фиксации подвижного объекта. С помощью кинематической модели решена задача нахождения баланса между минимальным временем, необходимым для совмещения осей подвижного объекта и пневматического схвата, и временем задержки, которое регулируется с помощью системного таймера. Представлены результаты анализа точности времени срабатывания исполнительных механизмов, которое оценивается регрессионным методом. Проведено экспериментальное исследование с фиксацией времени выполнения рабочего цикла. Представлен сравнительный анализ данных в виде таблицы для двух режимов управления. Среднее время выполнения составило 13,97 с, что повысило быстродействие работы МК на 7,97 с в сравнении с ручным режимом. Проверена надежность работы мехатронного комплекса, результаты представлены на основании выполнения 100 циклов.

Работа выполнена при поддержке Госзадания (Соглашение № 2.3440.2017/4.6) и гранта РНФ № 16-19-00186 (оценка точности кинематической модели в разд. "Экспериментальные исследования").

Список литературы

1. **Бобырь М. В., Дородных А. А., Якушев А. С.** Устройство и программная модель управления пневматическим мехатронным комплексом // Мехатроника, автоматизация, управление. 2018. Т. 19, № 9. С. 612—617. DOI:10.17587/mau.19.612-617.

2. **Beater P.** Pneumatic drives: system design, modelling and control. Berlin Heidelberg: Springer-Verlag, 2007. 324 p.

3. **Pfeffer A., Glück T., Schausberger F., Kugi A.** Control and estimation strategies for pneumatic drives with partial position information // Mechatronics. 2018. N. 50. P. 259—270. DOI: 10.1016/j.mechatronics.2017.09.012.

4. **Wang Y., Ying Y., Xie C., Wang H., Feng X.** Mechatronics education at CDHAW of Tongji University: Laboratory guidelines, framework, implementations and improvements // Mechatronics. 2009. N. 19. P. 1346—1352. DOI: 10.1016/j.mechatronics.2009.09.001.

5. **Saravanakumar D., Mohan B., Muthuramalingam T., Sakthivel G.** A review on recent research trends in servo pneumatic positioning systems // Precision Engineering. 2017. N. 49. P. 481—492.

6. **Юхимец Д. А., Юдинков Э. Э.** Разработка прикладного программного интерфейса для управления роботом-манипулятором Mitsubishi RV-2FB // Программная инженерия. 2018. Т. 9, № 6. С. 253—261. DOI:10.17587/prin.9.253-261.

7. **Свидетельство** 2019610810 РФ. Свидетельство об официальной регистрации программы для ЭВМ. Программа для управления автоматизированным мехатронным комплексом с помощью операторской панели / М. В. Бобырь, А. А. Дородных, А. С. Якушев; заявитель и правообладатель ФГБОУ ВО ЮЗГУ (RU). № 2018665329; заяв. от 29.12.18, опубл. 18.01.19.

8. **Backe W.** The application of servo-pneumatic drives for flexible mechanical handling techniques // Robotics. 1986. Vol. 2, N. 1. P. 45—56.

9. **Östring M., Gunnarson S., Norrlöf M.** Closed-loop identification of an industrial robot containing flexibilities // Control Engineering Practice. 2003. Vol. 11, N. 3. P. 291—300. DOI:10.1016/S0967-0661(02)00114-4.

10. **Arreguin J. M. R., Ortega J. C. P., Gorrostieta E., Troncoso R. J. R.** Artificial intelligence applied into pneumatic flexible manipulator // Proceedings of 7th Mexican International Conference on Artificial Intelligence 2008. P. 339—345. DOI:10.1109/MICAI.2008.76.

11. **Elgeneidy K., Loshe N., Jackson M.** Bending angle prediction of control soft pneumatic actuators with embedded flex sensor — A data driven approach // Mechatronics. 2018. N. 50, P. 234-237. DOI:10.1016/j.mechatronics.2017.10.005.

12. **Саламандра Б. Л.** Анализ способов стабилизации положения этикетки на автоматических упаковочных машинах // Проблемы машиностроения и надежности машин. 2017. № 2. С. 106—112.

13. **Шелехов В. И., Тумуров Э. Г.** Технология автоматного программирования на примере программы управления лифтом // Программная инженерия. 2017. Т. 8, № 3. С. 99—111. DOI:10.17587/prin.8.99-111.

14. **Bobyr M. V., Yakushev A. S., Dorodnykh A. A.** Fuzzy devices for cooling the cutting tool of the CNC machine implemented on FPGA // Measurement. 2020. Vol. 152. DOI: 10.1016/j.measurement.2019.107378.

15. **Круглова Т. Н.** Интеллектуальный метод диагностирования и прогнозирования технического состояния мехатронных комплексов, эксплуатируемых в экстремальных условиях // Мехатроника, автоматизация, управление. 2011. № 3. С. 47—51.

16. **Чунихин А. Ю., Глазунов В. А**. Разработка механизмов параллельной структуры с пятью степенями свободы, предназначенных для технологических роботов // Проблемы машиностроения и надежности машин. 2017. № 4. С. 3—11.

17. **Bobyr M. V., Milostnaya N. A., Kulabuhov S. A.** A method of defuzzification based on approach of areas' ratio // Applied soft computing. 2017. N. 59. P. 19—32. DOI:10.1016/j.asoc.2017.05.040.

18. **Лошицкий П. А., Шеховцова Е. Е.** Расчет и моделирование работы промышленного манипулятора на силовых оболочковых элементах // Мехатроника, автоматизация, управление. 2015. Т. 16, № 7. С. 470—475. DOI: 10.17587/mau.16.470-475.

19. **Скворцов А. А.** Применение конечных автоматов при разработке программного обеспечения станков со сложной структурой // Программная инженерия. 2018. Т. 9, № 7. С. 332—336. DOI:10.17587/prin.9.332-336.

Hardware-Software Mechatronic Complex for Fixing Moving Objects

M. V. Bobyr, fregat_mn@rambler.ru, **A. A. Dorodnykh,** alex.dorodnych@mail.ru,
A. S. Yakushev, alekseyakushev@yandex.ru, **V. A. Bulatnikov,** rb465687@gmail.com

Corresponding author:

Bobyr Mikhail V., Professor, South-West State University, Kursk, 305040, Russian Federation
E-mail: fregat_mn@rambler.ru

Received on November 15, 2019
Accepted on December 20, 2019

The device of the research of mechatronic complex for detection, fixation, storage of mobile objects, its structure and the principle of action are presented in the article. The main components of the mechatronic complex are: pneumatic actuators controlled by Siemens S7-1200 programmable logic controller; the system for detecting moving objects and controlling the movement of actuators in real time, consisting of magnetic position sensors and an optical sensor.

There are problems in the automatic control mode of the mechatronic complex. It is necessary to determine the balance between minimum time for combining the axes of a moving object and pneumatic actuators and delay time for implementing a command to fix a moving object. These process variables are controlled by system delay timers. Thus, the technological operation of fixing a moving object is the most complicated operation in the mechatronic complex for programming.

The authors have developed mathematical methods and software to solve problems of combining the axes, allowing to synchronize technological operations in two modes. The kinematic model of the mechatronic complex, which allows to determine delay time for fixing a moving object, is presented. The sequence of movements of the actuator mechanisms, depending on the signal states of the optical and magnetic position sensors, is depicted as a cyclogram and implemented in the language of relay-contact circuits. Usage of system timers in the structure of the software code, which ensure the combination of axes of pneumatic capture and the moving object, is the main feature of the mechatronic complex. Experimental results show an improvement in reliability and speed when mechatronic complex works in the automatic control mode.

Keywords: *mechatronic complex, program code, programmable logic controller, pneumatic actuators*

Acknowledgements: *This article was completed with the support of State assignment: Agreement № 2.3440.2017/4.6 and grant of Russian Science Foundation № 16-19-00186*

For citation:

Bobyr M. V., **Dorodnykh A. A.**, **Yakushev A. S.**, **Bulatnikov V. A.** Hardware-Software Mechatronic Complex for Fixing Moving Objects, *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 77—85

DOI: 10.17587/prin.11.77-85

References

1. **Bobyr M. V., Dorodnykh A. A., Yakushev A. S**. Device and Program Model of Pneumatic Mechatronic Complex Control, *Mekhatronika, avtomatizatsiya, upravlenie*, 2018, vol. 19, no. 9, pp. 612—617, DOI: 10.17587/mau.19.612-617 (in Russian).
2. **Beater P.** *Pneumatic drives: system design, modelling and control*, Berlin Heidelberg, Springer-Verlag, 2007, 324 p.
3. **Pfeffer A., Glück T., Schausberger F., Kugi A.** Control and estimation strategies for pneumatic drives with partial position information, *Mechatronics*, 2018, no. 50, pp. 259—270, DOI: 10.1016/j.mechatronics.2017.09.01.01.
4. **Wang Y., Ying Y., Xie S., Wang H., Feng X**. Mechatronics education at CDHAW of Tongji University: Laboratory guidelines, framework implementations and improvements, *Mechatronics*, 2009, no. 19. pp. 1346—1352, DOI: 10.1016/j.mechatronics.2009.09.001.
5. **Saravanakumar D., Mohan B., Muthuramalingam T., Sakthivel G.** A review on recent research trends in servo pneumatic positioning systems, *Precision Engineering,* 2017, no. 49, pp. 481—492.
6. **Yuhimets D. A., Yudinkov E. E.** Development of an application software interface for controlling a Mitsubishi RV-2FB robotic arm, *Programmnaya Ingeneria,* 2018, vol. 9, no. 6, pp. 253—261, DOI: 10.17587/prin.9.253-261 (in Russian).
7. **Svidetelstvo 2019610810 RF.** Svidetelstvo ob ofitsialnoy registratsii programmy dlya EVM. Programma dlya upravleniya avtomatizirovannym mekhatronnym kompleksom s pomoshchyu operatorskoy paneli / M. V. Bobyr. A. A. Dorodnykh. A. S. Yakushev; zayavitel i pravoobladatel FGBOU VO YuZGU (RU). № 2018665329; zayav.ot 29.12.18. opubl. 18.01.19 (in Russian).
8. **Backe W.** The application of servo-pneumatic drives for flexible mechanical handling techniques, *Robotics,* 1986, vol. 2, no. 1, pp. 45—56.
9. **Östring M., Gunnarson S., Norrlöf M.** Closed-loop identification of an industrial robot containing flexibilities, *Control Engineering Practice*, 2003, vol. 11, no. 3. pp. 291—300, DOI:10.1016/S0967-0661(02)00114-4.
10. **Arreguin J. M. R., Ortega J. C. P., Gorrostieta E., Troncoso R. J. R**. Artificial intelligence applied into pneumatic flexible manipulator, *Proceedings of 7th Mexican International Conference on Artificial Intelligence 2008*, 2008, pp. 339—345, DOI: 10.1109/MICAI.2008.76.
11. **Elgeneidy K., Loshe N., Jackson M.** Bending angle prediction of control soft pneumatic actuators with embedded flex sensor — A data driven approach, *Mechatronics,* 2018, no. 50, pp. 234—237.
12. **Salamandra B. L.** Analysis of label position stabilization methods on automatic packaging machines, *Journal of Machinery Manufacture and Reliability*, 2017, vol. 46, no. 2, pp. 181—186.
13. **Shelekhov V. I., Tumurov E. G.** Applying Automata-based Software Engineering for the Lift Control Program, *Programmnaya Ingeneria*, 2017, vol. 8, no. 2, pp. 99—111, DOI: 10.17587/prin. 8.99-111 (in Russian).
14. **Bobyr M. V., Yakushev A. S., Dorodnykh A. A.** Fuzzy devices for cooling the cutting tool of the CNC machine implemented on FPGA, Measurment, 2020, DOI: 10.1016/j.measurement.2019.107378
15. **Kruglova T. N.** Intellectual Method of Diagnosing and Forecasting of a Technical Condition of the Mechatronic Complexes for Extreme Conditions, *Mehatronika, avtomatizatsiya, upravlenie,* 2011, no. 3, pp. 47—51 (in Russian).
16. **Chunikhin A. Yu., Glazunov V. A.** Developing the mechanisms of parallel structure with five degrees of freedom designed for technological robots, *Journal of Machinery Manufacture and Reliability*, 2017, vol. 46, no. 4, pp. 313—321.
17. **Bobyr M. V., Milostnaya N. A., Kulabuhov S. A.** A method of defuzzification based on approach of areas' ratio, *Applied soft computing,* 2017, no. 59, pp. 19-32, DOI: 10.1016/j.asoc.2017.05.040.
18. **Loshitskiy P. A., Shehovcova E. E.** Calculation and Simulation of an Industrial Manipulator on the Power of the Shell Elements, *Mekhatronika, avtomatizatsiya, upravlenie*, 2015, vol. 16, no. 7, pp. 470—475, DOI: 10.17587/mau.16.470-475 (in Russian).
19. **Skvortsov A. A.** The Use of Finite State Machines in the Development of Software for Machines with Complex Structure, *Programmnaya Ingeneria*, 2018, vol. 9, no. 7, pp. 332—336, DOI: 10.17587/ prin.9.332-336 (in Russian).

УДК 004.8, 004.93 DOI: 10.17587/prin.11.86-95

Д. Д. Рухович, аспирант, daniel-rukhovich@yandex.ru, Московский государственный университет имени М. В. Ломоносова

Оценка абсолютного масштаба в монокулярных SLAM-системах с использованием синтетических данных

Предложены решения задачи неточной оценки масштаба в монокулярных SLAM-системах путем вычисления абсолютных движений камеры между соседними кадрами видеопоследовательности с использованием нейросетевых моделей. Описываемый метод оценки масштаба позволяет определять собственное движение камеры по кадрам, полученным с помощью одной монокулярной камеры. Предложенный метод превосходит по точности существующие классические и нейросетевые методы оценки собственного движения камеры. Он также может быть использован в качестве одного из компонентов классической SLAM-системы. В данной работе продемонстрирована возможность обучения нейросетевых моделей на синтетических данных, полученных из компьютерного симулятора. Приведены результаты экспериментов, показывающие, что использование только синтетических данных при обучении позволяет получить точность, сравнимую с точностью при обучении на реальных данных. Предложенный метод адаптирован для видеопоследовательностей с низким разрешением, что позволяет применять его в режиме реального времени.

Ключевые слова: *машинное обучение, глубокое обучение, SLAM, оценка масштаба*

Введение

Решение задачи SLAM (*Simultaneous Localization and Mapping*; задача одновременной локализации и построения карты) подразумевает определение собственного движения (поворота и смещения) камеры, а также построение карты сцены, по которой перемещается камера. Так как найденное SLAM-системой решение инвариантно к масштабу данных, линейные размеры построенной карты могут различаться на разных участках видеопоследовательности. Здесь и далее под масштабом трехмерной сцены подразумевается расстояние в метрах между двумя ее фиксированными точками. При фиксированных координатах всех точек известный масштаб позволяет вычислить расстояние между любой их парой.

В данной работе предложено решение этой задачи путем введения дополнительного ограничения на неизменность абсолютного масштаба данных внутри заданной видеопоследовательности. Задача рассматривается в наиболее общей постановке, т. е. без использования предположений о сцене или находящихся внутри нее объектах.

Для решения задачи оценки масштаба на практике используются дополнительные сенсоры, такие как инерциальные измерительные модули (IMU), системы глобального позиционирования (GPS, ГЛОНАСС), лидары, стереокамеры, сгруппированные камеры [1] или камеры глубины, оценивающие расстояния до объектов сцены. Однако они повышают стоимость, сложность, энергопотребление и массу всей системы в целом, тем самым ограничивая возможность практического применения. Таким образом, задача построения метода оценки масштаба по данным с единственной камеры является актуальной.

Достижения последних лет в области глубинного обучения привели к развитию новых методов решения задачи SLAM. Например, в некоторых современных SLAM-системах используются нейросетевые методы определения глубины сцены по кадру с единственной камеры. Это позволяет оценить абсолютный масштаб сцены и таким образом повысить качество работы SLAM-системы в целом [2, 3]. Однако подобные SLAM-системы решают более сложную задачу и потому не только являются более ресурсоемкими, но и требуют большего объема данных для обучения. В последнее время также ведется работа по созданию SLAM-системы полностью на основе глубокого обучения [4]. Замена эвристических компонентов SLAM-системы на обучаемые позволила бы, в частности, оценивать абсолютный масштаб при условии того, что в обучающей выборке собственное движение камеры также описано в абсолютных значениях. Однако в настоящее время обучаемые методы решения задачи SLAM не позволяют добиться точности, которую демонстрируют классические методы. Они требуют больших вычислительных ресурсов, что ограничивает возможности их применения на практике.

Метод оценки масштаба, рассматриваемый в настоящей работе, опирается на метод, предложенный в работе [5]. В данном методе оценка масштаба реализуется через вычисление абсолютных расстояний между положениями камеры для подряд идущих кадров видеопоследовательности, причем каждая пара кадров рассматривается независимо от остальных.

Такой метод не использует предположения о геометрии сцены и находящихся в ней объектов, и потому может быть легко интегрирован в существующие SLAM-системы.

Детальный анализ результатов работы SLAM-системы [5] на наборе данных KITTI [6] обнаруживает ряд проблемных вопросов. Во-первых, наблюдается систематическая погрешность оценки масштаба в точках поворота траектории камеры, вызванная преобладанием прямых участков траекторий в обучающей выборке. Во-вторых, вариативность комбинаций поворотов и смещений в обучающей выборке напрямую зависит от расположения камеры относительно центра вращения несущего ее транспортного средства. Например, у автомобиля, использованного для сбора данных KITTI, глобальный центр вращения расположен на задней оси [7]. Это означает, что поворот камеры, размещенной в передней части автомобиля, всегда будет сочетаться с ненулевым поступательным движением. Адаптация модели к новым конфигурациям камер потребует сбора дополнительных данных. Последним вопросом является недостаточная точность базового метода: полученные с его помощью оценки пройденного расстояния между последовательными кадрами значительно различаются даже при постоянной скорости транспортного средства. Учитывая перечисленные недостатки выбранного метода, целевые установки исследования могут быть сформулированы следующим образом:

1) повышение качественных характеристик выбранного базового метода, а именно точности оценки абсолютного масштаба;

2) уменьшение погрешности оценки масштаба в точках поворота траектории камеры;

3) снижение зависимости качества работы метода от конфигурации установки для сбора данных, а именно от изменения расположения камеры относительно несущего ее транспортного средства.

Для решения первой и второй задач предлагается изменить архитектуру нейронной сети, предложенной в работе [5], путем ее незначительного усложнения и добавления рекуррентного модуля. Экспериментальная проверка показала, что подобные изменения позволяют существенно повысить точность. Возможности дальнейшего улучшения архитектуры нейронной сети не являются предметом рассмотрения в данной работе.

Необходимость решения третьей задачи возникает в силу недостаточного разнообразия данных в обучающей выборке. Она может быть решена с помощью симуляторов автономного вождения, которые позволяют легко моделировать любой тип конфигурации камеры и генерировать большое количество обучающих данных с широким диапазоном условий освещения и погоды. На настоящее время синтетические данные успешно применяют для решения задач в смежных областях (для оценки глубины [8] и восстановления оптического потока [9]), что дает основание полагать, что их использование для решения задачи SLAM окажется не менее продуктивным. Несмотря на то что симулируемое окружение значительно отличается от сцен KITTI [6] (например, отсутствием припаркованных автомобилей), предложенный метод демонстрирует способность к обобщению на непредставленные в обучающей выборке сцены. В разделе, посвященном экспериментам, показано, что обучение на синтетических данных позволяет добиться точности, сравнимой с точностью, достигаемой при обучении на реальных данных. Кроме того, исследуется зависимость между степенью фотореалистичности синтетических изображений и точностью решения поставленной задачи.

Обзор публикаций по теме

Настоящий раздел содержит краткое описание принципов работы и особенностей существующих методов оценки абсолютного масштаба для монокулярных SLAM-систем со ссылками на соответствующие публикации. Эти методы можно разделить на три группы по критерию использования информации о сцене:

1) использующие информацию о сцене;

2) использующие ограничивающие предположения о сцене;

3) общие, т. е. не использующие информацию или ограничивающие предположения о сцене.

Методы первой группы используют наиболее полную информацию о сцене, например, в форме трехмерной модели сцены с известными абсолютными расстояниями между точками. Для оценки масштаба построенной сцены используются попарные соответствия между пикселями изображений и точками этой модели. Однако трехмерная модель сцены, известная наперед, является очень сильным допущением, которое сильно ограничивает применимость этих методов на практике. Эти методы впервые были описаны более 30 лет назад [10]. На настоящее время они имеют исключительно историческую ценность и упоминаются здесь, в первую очередь, для полноты картины.

В методах второй группы вводятся ограничивающие предположения о сцене. Например, некоторые методы этой группы определяют высоту камеры относительно поверхности, по которой перемещается агент, если известно, что эта поверхность является плоской и выполнено условие неголономности: камера жестко зафиксирована, а ее высота постоянна. Эти предположения верны для некоторых колесных транспортных средств (автомобили, велосипеды), но не выполняются для агентов, совершающих движения в трех плоскостях (летательные аппараты), или в тех случаях, когда камера закреплена на гибком штативе. Таким образом, указанные методы могут применяться для решения задачи SLAM в определенных сценариях, что было продемонстрировано в работах [11—14], однако универсальными не являются.

Другие методы [7, 15—18] второй группы оценивают размеры некоторых объектов сцены, используя вспомогательную модель — предобученный детектор объектов некоторого множества классов (например, автомобилей). Однако объекты, размеры которых

можно оценить с помощью детектора, могут отсутствовать в части сцен. Более того, эти объекты могут быть видны только частично, что затрудняет оценивание их размеров и является потенциальным источником ошибок. В ряде работ были предложены нетривиальные схемы, позволяющие обойти некоторые ограничения подобных методов. Например, в работе [16] рассматривался алгоритм глобальной оптимизации, использующий координаты трехмерных точек, ассоциированных с объектами известного размера, а в работе [19] к совокупности детекций объектов применялся общий байесовский вывод.

Третья группа методов является общей, поскольку не использует точной информации или предположений о сцене. Большинство подобных методов [20, 21] строит трехмерную модель сцены на основе информации об абсолютной глубине сцены. Глубина сцены может быть непосредственно измерена сенсорами или же приблизительно определена с помощью нейросетевых моделей. Прогресс по оценке глубины, достигнутый в последних работах [8, 22, 23], позволяет применять такие модели к видеопоследовательностям с монокулярной камеры и по полученному приближению определять абсолютный масштаб сцены с достаточной точностью.

В другом методе, описанном в работе [5], нейросетевая модель обучается предсказывать абсолютное расстояние, пройденное камерой, для пар кадров со значительным визуальным перекрытием. Полученные значения используются затем в глобальной оптимизации сцены, что существенно повышает точность оценки абсолютного масштаба. Метод спроектирован для работы с изображениями в низком разрешении (240 × 120 пикселей). Для их обработки требуется небольшое количество вычислительных ресурсов, и потому этот метод может применяться в режиме реального времени. Авторы этой работы описывают преимущества интеграции модели оценки абсолютного масштаба в монокулярную SLAM-систему. Обучение и применение нейросетевых моделей на синтетических данных вместо реальных возможно только при достижении определенного сходства между данными, принадлежащими этим двум доменам. В последнее время было представлено большое число методов доменной адаптации без учителя, которые позволяют повысить визуальное сходство изображений из разных доменов. Тем не менее неясно, насколько важна фотореалистичность изображений для решения конкретной задачи. Для исследования этого вопроса в данной работе использованы различные обучаемые методы доменной адаптации [24, 25]. Эти методы обучались совместно с методом оценки масштаба — таким образом были определены преобразования синтетических изображений, позволяющие повысить точность решения целевой задачи.

Предлагаемый метод

Настоящий раздел состоит из двух частей, посвященных соответственно процедуре сбора синтетических данных и архитектурам нейронных сетей.

Синтетические данные

Сбор больших реальных видеоданных является ресурсоемкой процедурой, потому он обычно сильно ограничен во времени и пространстве, что является причиной недостатков, присущих многим существующим наборам реальных данных. Во-первых, в силу временных рамок вариативность погодных условий, освещенности и времени суток может быть невелика (рис. 1, см. третью сторону обложки).

Во-вторых, собранные данные часто имеют локальную специфику. Например, городские сцены значительно отличаются от сельских, причем не только по общему визуальному впечатлению, но и по классам объектов, присутствующих в сцене, и по среднему числу этих объектов. Кроме того, расположение камеры на транспортном средстве ограничивает изменчивость наблюдаемых данных, а изменение ее расположения может привести к ошибкам в работе метода, так как в обучающей выборке не представлены некоторые комбинации смещения и поворота. Таким образом, для обучения универсального и робастного метода понадобится большой и достаточно разнообразный набор данных, сбор которых требует значительных средств и усилий. Одним из способов решения этой проблемы является использование синтетических данных из симуляторов автономного вождения, таких как [26—28]. Симулятор позволяет прикрепить одну или несколько камер в определенных местах на транспортном средстве, изменять погоду, время суток и условия освещения, а также варьировать собственные параметры камеры. Режим автопилота позволяет собирать данные в автоматическом режиме. Как будет показано ниже, использование данных, полученных таким образом, позволяет достичь точности оценки масштаба, сопоставимой с точностью, достигаемой при обучении на реальных данных.

Архитектуры нейронных сетей

Метод, описанный в работе [5], выбран в исследовании, представленном в данной работе, в качестве базового. Этот метод устроен следующим образом. На вход нейросетевой модели в момент времени i подается пара кадров в формате RGB с номерами i и $i + 1$, объединенных в одно шестиканальное изображение. Далее нейросетевая модель извлекает признаки из полученных изображений. Для этого последовательно применяют три вычислительных блока, каждый из которых состоит из одного сверточного слоя и одного субдискретизирующего слоя, уменьшающего размер входного тензора в 2 раза по обеим пространственным осям. Извлеченные таким образом признаки поступают на вход блоку, состоящему из двух полносвязных слоев, который решает задачу регрессии для шести степеней свободы камеры. Простота этой нейросетевой архитектуры потенциально может ограничивать предсказательную силу модели. Как следствие, так как одной из целей данной работы является повышение качества работы метода, первым шагом в этом направлении становится увеличение сложности архитектуры. Добавление всего двух сверточных слоев (как показано на рис. 2,

см. третью сторону обложки) позволяет значительно увеличить точность.

На рис. 2 сверточные (*convolutional*) слои обозначены *conv*1, ..., *conv*5, субдискредитирующие (*max pooling*) — max pool2 × 2 и полносвязные (*fully connected*) — *FC*1 и *FC*2. Над сверточными и полносвязными слоями приведены размеры их выходов. $D_{i,i+1}$ обозначает предсказываемое расстояние между положениями камеры в моменты времени i и $i + 1$.

Проведенные эксперименты показали, что замена функции активации сверточных слоев на экспоненциально-линейную функцию приводит к ускорению сходимости обучения. Также для предотвращения переобучения после каждого сверточного и полносвязного слоя был добавлен исключающий регуляризационный слой. Подробные параметры конфигурации всех сверточных слоев указаны в табл. 1. В дальнейшем описанная архитектура будет обозначаться как CNN (*Convolutional Neural Network*).

Предсказания существующих методов оценки движения камеры по отдельным парам кадров нередко имеют значительный разброс в пределах небольших отрезков времени. Этот недостаток подобных методов является следствием того, что при оценке пройденного расстояния не учитывается модель движения транспортного средства (допустимый диапазон скоростей, ускорения и торможения, инерция и т. д.). Одним из возможных решений является добавление ограничений на предсказанные движения транспортного средства в явном виде. Однако нейросетевые модели потенциально способны вывести эти ограничения напрямую из обучающих данных. В настоящей работе для этого используются рекуррентные нейросетевые модели.

Рекуррентная модель отличается от сверточной тем, что поддерживает память о своих скрытых состояниях во времени и имеет среди них циклы обратной связи, что позволяет ее текущему скрытому состоянию являться функцией предыдущих, как показано на рис. 2 (см. третью сторону обложки). Таким образом, рекуррентная модель может обнаруживать взаимосвязи между текущим входом и предыдущими состояниями последовательности. По заданному вектору признаков x_i в момент времени i рекуррентный модуль обновляется по формулам

$$h_i = H\left(W_{hx}x_i + W_{hh}h_{i-1} + b_h\right),$$

Таблица 1

Конфигурация слоев сверточной сети (CNN)

Номер слоя	Размер ядра, пиксели	Размер отступа, пиксели	Число фильтров
1	11×11	5	32
2	9×9	4	64
3	7×7	3	128
4	5×5	2	256
5	3×3	1	512

$$y = W_{hy}h_i + b_y,$$

где h_i и y_i — скрытое состояние и выходной вектор в момент времени i соответственно; W — соответствующие матрицы весов; b — векторы сдвигов; H — поэлементная нелинейная функция активации, например, сигмоида или гиперболический тангенс. Хотя теоретически рекуррентные модели могут обучаться на последовательностях произвольной длины, на практике эти длины ограничены из-за известной проблемы затухания градиента [29].

В данной работе в качестве конкретной реализации рекуррентного модуля используется долгая краткосрочная память (LSTM; *Long Short-Term Memory*) [30], которая позволяет обучаться долговременным зависимостям, вводя четыре вентиля памяти. Развернутая структура модуля LSTM показана на рис. 3.

По заданному вектору признаков x_i в момент времени i и вектору скрытого состояния h_{i-1} параметры модуля LSTM обновляются по формулам

$$j_i = \sigma_s\left(W_{xj}x_i + W_{hj}h_{i-1} + b_j\right),$$

$$f_i = \sigma_s\left(W_{xf}x_i + W_{hf}h_{i-1} + b_f\right),$$

$$g_i = \sigma_t\left(W_{xg}x_i + W_{hg}h_{i-1} + b_g\right),$$

$$c_i = f_i \odot c_{i-1} + j_i \odot g_i,$$

$$o_i = \sigma_s\left(W_{xo}x_i + W_{ho}h_{i-1} + b_o\right),$$

$$h_i = o_i \odot \sigma_t\left(c_i\right),$$

где $\odot$ — операция поэлементного умножения; σ_s — сигмоидная функция активации; σ_t — гиперболический тангенс; W — соответствующие матрицы весов; b — векторы сдвигов; j_i, f_i, g_i, o_i — векторы вентилей входа, забывания, модуляции и выхода соответственно; c_i — вектор памяти в момент времени i.

Добавление модуля LSTM позволяет нейросетевой модели накапливать информацию о движениях между предыдущими парами кадров. Идея использования рекуррентного модуля восходит к работе [31], где LSTM применяется для распознавания действий по видеопоследовательности. Впервые LSTM-модуль

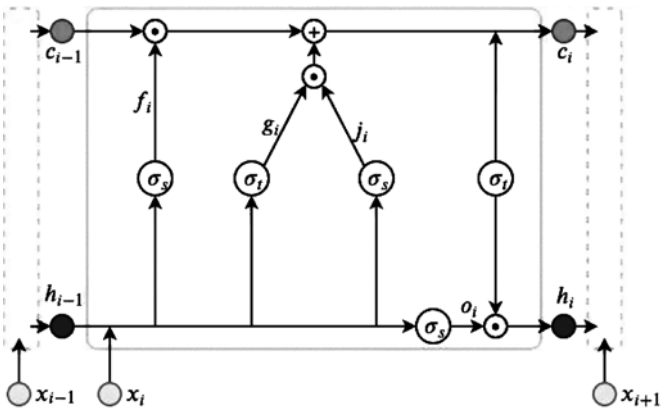

Рис. 3. Структура LSTM-модуля: ⊙ и ⊕ обозначают поэлементное умножение и сложение двух векторов соответственно

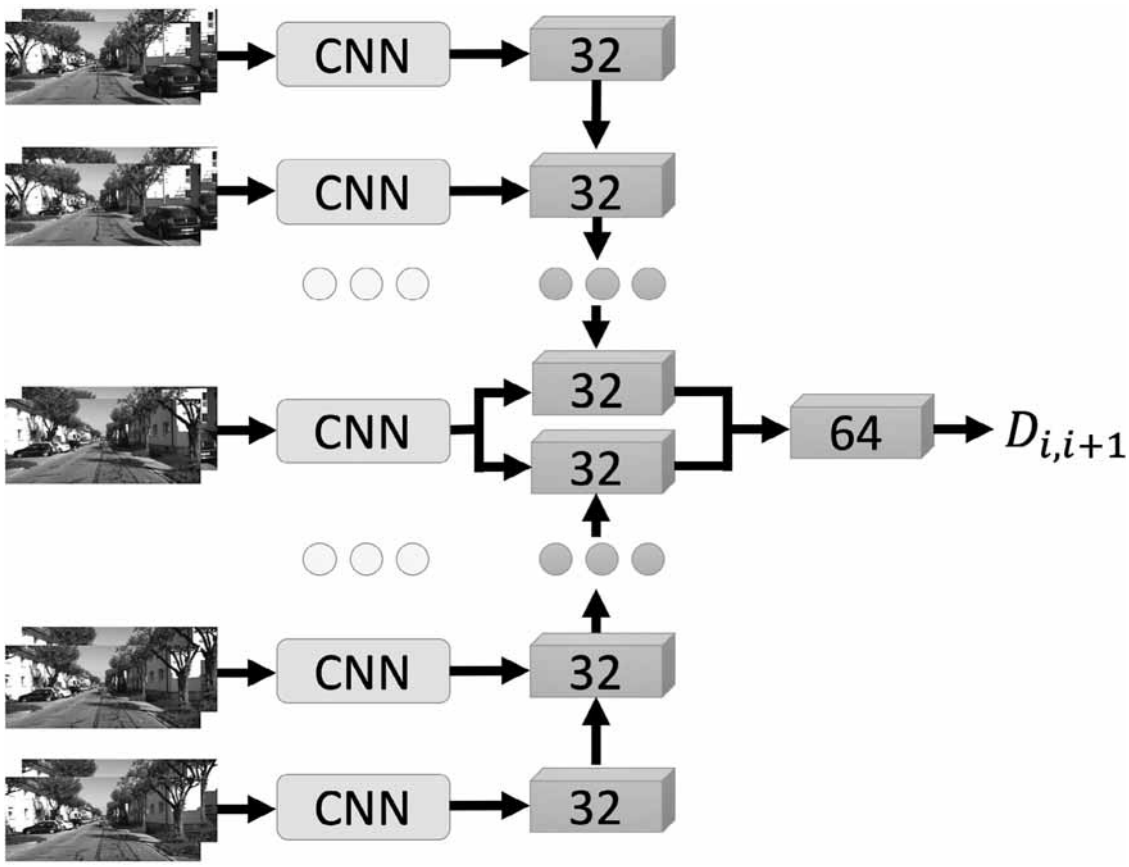

Рис. 4. Архитектура сети с рекуррентным модулем. Сверточная часть (CNN) соответствует зеленому блоку на рис. 2, см. третью сторону обложки

был использован для решения задачи оценки собственного движения камеры в работе [4]. В отличие от существующих методов, в данной работе применяется также двунаправленная версия LSTM, которая использует информацию не только с предыдущих, но и с нескольких следующих кадров (рис. 4).

При таком подходе предсказания совершаются с некоторой задержкой. Однако вычисления внутри LSTM могут выполняться параллельно с другими шагами SLAM-системы, такими как извлечение ключевых точек и вычисление их дескрипторов. Кроме того, операции внутри рекуррентного модуля являются легковесными, поэтому потери производительности относительно базовой модели CNN оказываются незначительными. Предлагаемые изменения в базовой архитектуре [5] позволяют добиться ошибки на обучающей выборке в пределах погрешности измерений данных. Следовательно, для решения поставленной задачи не требуется дальнейшее усложнение архитектуры.

Эксперименты

В этом разделе сначала описаны используемые реальные и синтетические наборы данных, а также параметры обучения предложенных нейросетевых моделей. Затем, для оценивания предлагаемых методов приведена следующая серия экспериментов:

- сравнение точности оценки абсолютного масштаба базовым методом [5] и с использованием предлагаемых CNN- и LSTM-моделей, обученных на реальных и/или синтетических данных;
- исследование размера истории LSTM;
- исследование важности разнообразия синтетических данных;
- исследование применимости доменной адаптации.

Эти результаты дополнены детальным анализом ошибок и их распределения для лучшей модели.

Процедура обучения

Для создания и обучения нейросетевых моделей использовалась библиотека глубокого обучения Tensorflow [32].

В базовом методе предлагалось использовать следующие параметры калибровки: фокусное расстояние установлено равным 250 пикселям, а координаты центральной точки по оси x и y — 140 и 60 пикселей соответственно. При этом размер изображения был увеличен с 240×120 до 280×120 пикселей.

В обучающую выборку были включены все пары кадров, где пройденное камерой расстояние между первым и вторым кадром не превышало $D_{\max} = 1{,}7$ м. Кроме того, пары кадров, записанные во время поворота транспортного средства, были повторно добав-

лены в обучающую выборку, чтобы отчасти скомпенсировать недостаток поворотов при преобладающих прямолинейных движениях.

Во время обучения к парам кадров применяли случайные аугментации — изменения изображений в целях увеличения размеров обучающей выборки, повышения робастности метода, а также борьбы с переобучением. Были использованы следующие аугментации:

- случайное изменение контрастности и яркости изображения;
- горизонтальное отражение изображения, применявшееся с вероятностью 0,5;
- поворот изображений на не более чем 10°;
- сдвиг всех пикселей изображения по горизонтальной или вертикальной оси в пределах 10 % от ширины и высоты изображения соответственно.

К кадрам одной пары применяли одинаковое случайное преобразование.

Сравнение с базовым методом [5] проводили на наборе данных KITTI [6] согласно протоколу, предложенному авторами этого метода. Из 12 видеопоследовательностей этого набора последовательности 01, 03, 04, 05, 06, 07, 09 вошли в обучающую выборку, а 00, 02 и 08 были использованы для тестирования. Таким образом, обучающая выборка содержала около 100 000 изображений, а тестовая — 26 000.

Для генерации синтетических видеопоследовательностей применяли симулятор CARLA [26], выбранный в силу наличия режима автопилота. Другие симуляторы, например, [27] или [28], могут быть использованы вместо CARLA или в дополнение к нему для дальнейшего увеличения разнообразия выборки. Симулятор CARLA позволяет изменять время суток и погодные условия, такие как наличие луж или интенсивность дождя, и предоставляет карты шести различных городов. Эти карты отличаются не только сетью дорог, но и визуальными образами представленных объектов. Процедура сбора синтетических видеопоследовательностей состоит из запуска транспортного средства с камерой в режиме автопилота на небольшое расстояние (около 100 м) со случайной стартовой позиции, причем погодные условия и время суток также выбирают случайно и независимо. Параметры камер выбирают такими же, как и в наборе данных KITTI. В сгенерированную таким образом итоговую выборку вошло около 800 000 кадров, собранных с шести карт.

Во всех экспериментах для обучения нейронной сети используется метод оптимизации Adam [33], минимизирующий среднее квадратичное отклонение путем стохастического градиентного спуска. Процедура обучения сверточной модели (CNN) была ограничена 100 000 итераций, скорость обучения изначально принималась равной 0,0001 и уменьшалась вдвое каждые 10 000 итераций, один пакет градиентного спуска состоял из 75 обучающих примеров, а вероятность отсева параметров в исключающих регуляризационных слоях устанавливалась равной 15 %. Сверточная часть рекуррентной модели LSTM была проинициализирована значениями весов модели CNN. После этого модель LSTM обучалась на протяжении 15 000 итераций, ее начальная скорость обучения полагалась равной 0,00002 и уменьшалась в 2 раза каждые 2500 итераций, а размер пакета градиентного спуска был равен 16.

Результаты

В качестве метрики для оценивания работы метода используется среднее квадратичное отклонение между предсказанными расстояниями и истинными. В табл. 2 приведено сравнение полученных результатов с результатами существующих методов.

Таблица 2

Сравнение результатов оценки абсолютного масштаба базового метода [5] и предложенных архитектур. В качестве рекуррентного модуля использован двунаправленный LSTM с размером истории 19

№	Метод	Обучающая выборка	Среднее квадратичное отклонение, м		
			KITTI #00	KITTI #02	KITTI #08
1	Из работы [12]	KITTI	0,252	0,160	0,349
2	Из работы [5]	KITTI	0,177	0,203	0,165
3	Базовый (240 × 120)	KITTI	0,177	0,180	0,152
4	Базовый (280 × 120)	KITTI	0,175	0,178	0,149
5	CNN	KITTI	0,107	0,113	0,092
6	CNN	CARLA	0,111	0,105	0,121
7	CNN	KITTI + CARLA	0,079	0,084	0,081
8	CNN + LSTM	KITTI	**0,069**	**0,083**	**0,064**
9	CNN + LSTM	CARLA	0,102	0,132	0,128
10	CNN + LSTM	KITTI + CARLA	0,076	0,084	0,084

Для каждой из трех тестовых видеопоследовательностей приведены результаты среднего квадратичного отклонения предсказанного и истинного значений в метрах.

В строках 1 и 2 табл. 2 приведены результаты существующих методов. В связи с тем что аугментация данных и параметры обучения, используемые в данной работе, отличаются от тех, что были использованы в работе [5], в строках 3 и 4 приведены результаты воспроизведения базовой архитектуры с этими отличиями. Предложенные изменения улучшают итоговую точность. Изменение внутренних параметров не оказывает значительного влияния на итоговые результаты (см. строки 3 и 4), что свидетельствует о том, что высокая точность может быть достигнута даже при меньшем размере входного изображения.

Результат в строке 5 подтверждает значительное улучшение метрики по сравнению с результатами [5] для модели, использующей архитектуру CNN и только реальные данные KITTI при обучении. Важным наблюдением являются небольшие отличия в значениях метрик при обучении только на реальных данных KITTI (строка 5) и только синтетических данных из симулятора (строка 6). Впрочем, лучшие значения метрик достигаются при использовании реальных и синтетических изображений (строка 7), что говорит в пользу комплиментарности этих двух доменов данных.

Наименьшее значение среднего квадратичного отклонения достигается при использовании архитектуры с двунаправленным LSTM-слоем с размером истории, равным 19 (строка 8, полужирный шрифт), что подтверждает важность учета динамики транспортного средства при обучении. Тем не менее предложенная рекуррентная архитектура не дает улучшений в случае обучения на синтетических данных из CARLA (строки 9, 10). Эти результаты можно объяснить отличием синтетических видеопоследовательностей от реальных: они очень гладкие и равномерные, в то время как реальные — более шумные и менее линейные.

Сравнение различных размеров истории LSTM в одно- и двунаправленной версии дано в табл. 3.

Таблица 3

Результаты однонаправленной (1) и двунаправленной (2) версий LSTM с различным размером истории на тестовых видеопоследовательностях KITTI #00, #02, #08

Версия LSTM	Размер истории	Среднее квадратичное отклонение, м
1	5	0,087
1	11	0,085
1	19	0,088
2	5	0,084
2	11	0,077
2	19	**0,076**

Таблица 4

Результаты предложенного метода на тестовых видеопоследовательностях KITTI #00, #02, #08 в зависимости от числа карт симулятора, используемых в обучающей выборке

Карты CARLA	Среднее квадратичное отклонение, м
1	0,139
1, 2	0,137
1, 2, 3	0,128
1, 2, 3, 4	0,129
1, 2, 3, 4, 5	0,129
1, 2, 3, 4, 5, 6	**0,115**

Эксперименты показывают, что этот размер не играет существенной роли в обоих случаях.

В рамках данной работы была проверена гипотеза о важности разнообразия синтетических данных для решения поставленной задачи. В табл. 4 приведены результаты предложенного метода в зависимости от числа карт городов симулятора, использовавшихся в обучающей выборке.

Следует отметить значительное повышение точности при добавлении в обучающую выборку видеопоследовательностей из виртуальных сцен, сильно отличающихся от уже в ней имеющихся. Так, карты городов 1 и 2 различаются незначительно и, соответственно, похожи результаты на тестовых последовательностях при использовании карт только города 1 и карт городов 1 и 2, вместе взятых. В карте города 3 появляются многополосные дороги, и добавление карт этого города при обучении уменьшает среднее квадратичное отклонение на тестовой выборке примерно на 1 см. Похожее улучшение происходит и при добавлении карты города 6 с автострадами.

В ходе исследования было изучено влияние степени фотореалистичности синтетических данных, используемых при обучении, на значения целевых метрик. В табл. 5 приведены результаты метода без доменной адаптации, с использованием методов T2Net [25] и CyCADA [24]. Эксперименты не показывают улучшения целевой метрики при использовании рассмотренных методов доменной адаптации. Таким образом, подтверждается вывод из работы [34]

Таблица 5

Результаты на тестовых последовательностях набора KITTI #00, #02, #08 для различных вариантов доменной адаптации

Метод	Среднее квадратичное отклонение, м
Без доменной адаптации	**0,115**
T2Net [25]	0,117
CyCADA [24]	0,120

о том, что фотореалистичность синтетических данных оказывается менее важна для решения поставленной задачи, чем их разнообразие.

Сравнение результатов базового [5] метода и лучшей из предложенных в этой работе моделей приведено на рис. 5, см. четвертую сторону обложки. Тестовые траектории наложены друг на друга, каждая точка обозначает положение камеры в некоторый момент времени, а цвет этой точки описывает среднюю квадратичную ошибку предсказания. Заметны оба преимущества предложенного метода, а именно: улучшение точности оценки абсолютного масштаба и отсутствие больших отклонений в точках поворота транспортного средства.

На рис. 6 (см. четвертую сторону обложки) приведены примеры кадров, на которых предложенная модель достигает самых больших ошибок в тестовой выборке. Их объединяет большой процент зеленой растительности относительно всей площади кадра, что затрудняет работу предложенного метода оценки абсолютного масштаба.

Заключение

Была рассмотрена задача оценки масштаба для монокулярных SLAM-систем путем определения расстояния между центрами камер последовательных кадров видеопоследовательности. Такая оценка позволяет улучшить общую точность SLAM-системы и выдавать трехмерную реконструкцию сцены с метрическими расстояниями между точками. Предложенный метод работает с каждой парой кадров независимо (или с небольшим числом соседних пар в случае использования рекуррентного модуля), что позволяет ему избежать накопления ошибки по всей траектории движения транспортного средства с камерой. Предложенные нейросетевые модели позволяют достичь существенно лучшей точности оценки абсолютного масштаба по сравнению с существующими методами. Особенно сильно точность предложенного метода превосходит базовый [5] в точках поворота траектории камеры. Также в работе рассмотрена возможность обучения модели только на синтетических данных из симулятора автономного вождения и показано, что при этом может быть достигнута точность, не отличающая от достигаемой при обучении на реальных данных, что является практическим решением проблемы реконфигурации камер под каждую новую задачу. Эксперименты с доменной адаптацией без учителя показывают, что для обучения методов оценки абсолютного масштаба на синтетических данных проблема их недостаточной фотореалистичности не актуальна. Предложенный метод адаптирован для видеопоследовательностей с низким разрешением, что позволяет применять его на практике в режиме реального времени.

Список литературы

1. **Шокуров А. В., Кривчикова К. А.** Оценка абсолютной погрешности при поиске внешних параметров группы видеокамер в приложении к задаче локализации и сопровождения объекта // Программная инженерия. 2017. Т. 8, № 10. С. 470—480.

2. **Tateno K., Tombari F., Laina I., Navab N.** CNN-SLAM: Real-time dense monocular slam with learned depth prediction // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 6243—6252.

3. **Yang N., Wang R., Stuckler J., Cremers D.** Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry // Proceedings of the European Conference on Computer Vision (ECCV). 2018. P. 817—833.

4. **Wang S., Clark R., Wen H., Trigoni N.** Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks // 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017. P. 2043—2050.

5. **Frost D., Murray D., Prisacariu V.** Using learning of speed to stabilize scale in monocular localization and mapping // 2017 International Conference on 3D Vision (3DV). IEEE, 2017. P. 527—536.

6. **Geiger A., Lenz P., Urtasun R.** Are we ready for autonomous driving? The kitti vision benchmark suite // 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012. P. 3354—3361.

7. **Scaramuzza D., Fraundorfer F., Pollefeys M., Siegwart R.** Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints // 2009 IEEE 12th International Conference on Computer Vision. IEEE, 2009. P. 1413—1419.

8. **Atapour-Abarghouei A., Breckon T. P.** Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. P. 2800—2810.

9. **Dosovitskiy A., Fischer P., Ilg E.** et al. Flownet: Learning optical flow with convolutional networks //Proceedings of the IEEE international conference on computer vision. 2015. P. 2758—2766.

10. **Rosenholm D. A. N., Torlegar D. K.** Three-dimensional absolute orientation of stereo models using digital elevation models // Photogrammetric engineering and remote sensing. 1988. Vol. 54, N. 10. P. 1385—1389.

11. **Geiger A., Lauer M., Wojek C.** et al. 3d traffic scene understanding from movable platforms // IEEE transactions on pattern analysis and machine intelligence. 2013. Vol. 36, N. 5. P. 1012—1025.

12. **Geiger A., Ziegler J., Stiller C.** Stereoscan: Dense 3d reconstruction in real-time // 2011 IEEE intelligent vehicles symposium (IV). IEEE, 2011. P. 963—968.

13. **Gräter J., Schwarze T., Lauer M.** Robust scale estimation for monocular visual odometry using structure from motion and vanishing points // 2015 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2015. P. 475—480.

14. **Zhou D., Dai Y., Li H.** Ground-Plane-Based Absolute Scale Estimation for Monocular Visual Odometry //IEEE Transactions on Intelligent Transportation Systems. 2019. Vol. 21. P. 791—802.

15. **Castle R., Gawley D., Klein G., Murray D. W.** Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras // Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007. P. 4102—4107.

16. **Frost D., Kähler O., Murray D.** Object-aware bundle adjustment for correcting monocular scale drift // 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016. P. 4770—4776.

17. **Kitt B., Geiger A., Lategahn H.** Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme // 2010 IEEE Intelligent vehicles symposium. IEEE, 2010. P. 486—492.

18. **Song S., Chandraker M.** Robust scale estimation in real-time monocular SFM for autonomous driving // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. P. 1566—1573.

19. **Sucar E., Hayet J. B.** Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift //2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2018. P. 1—7.

20. **Izadi S., Kim D., Hilliges O.** et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera // Proceedings of the 24th annual ACM symposium on User interface software and technology. 2011. P. 559—568.

21. **Kerl C., Sturm J., Cremers D.** Robust odometry estimation for RGB-D cameras // 2013 IEEE international conference on robotics and automation. IEEE, 2013. P. 3748—3754.

22. **Eigen D., Puhrsch C., Fergus R.** Depth map prediction from a single image using a multi-scale deep network // Advances in neural information processing systems. 2014. P. 2366—2374.
23. **Fu H., Gong M., Wang C.** et al. Deep ordinal regression network for monocular depth estimation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. P. 2002—2011.
24. **Hoffman J., Tzeng E., Park T.** et al. Cycada: Cycle-consistent adversarial domain adaptation // arXiv preprint arXiv:1711.03213. 2017.
25. **Zheng C., Cham T., Cai J.** T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks // Proceedings of the European Conference on Computer Vision (ECCV). 2018. P. 767—783.
26. **Dosovitskiy A., Ros G., Codevilla F.** et al. CARLA: An open urban driving simulator // arXiv preprint arXiv:1711.03938. 2017.
27. **Quiter C., Ernst M.** deepdrive/deepdrive: 2.0. 2018. URL: https://github.com/deepdrive/deepdrive
28. **Shah S., Dey D., Lovett C., Kapoor A.** Airsim: High-fidelity visual and physical simulation for autonomous vehicles // Field and service robotics. Springer, Cham, 2018. P. 621—635.
29. **Goodfellow I., Bengio Y., Courville A**. Deep learning. MIT press, 2016. 776 p.
30. **Hochreiter S., Schmidhuber J.** Long short-term memory // Neural computation. 1997. Vol. 9, N. 8. P. 1735—1780.
31. **Donahue J., Hendricks L. A., Rohrbach M.** et al. Long-term recurrent convolutional networks for visual recognition and description //Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. P. 2625—2634.
32. **Abadi M., Agarwal A., Barham P.** et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems // arXiv preprint arXiv:1603.04467. 2016.
33. **Kingma D. P., Ba J.** Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. 2014.
34. **Mayer N., Ilg E., Fischer P.** et al. What makes good synthetic training data for learning disparity and optical flow estimation? // International Journal of Computer Vision. 2018. Vol. 126, N. 9. P. 942—960.

Estimation of Absolute Scale in Monocular SLAM Using Synthetic Data

D. D. Rukhovich, daniel-rukhovich@yandex.ru, Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119234, Russian Federation

Corresponding author:

Rukhovich Danila D., Postgraduate Student, Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119234, Russian Federation
E-mail: daniel-rukhovich@yandex.ru

Received on January 17, 2020
Accepted on February 10, 2020

This paper addresses the problem of scale estimation in monocular SLAM by estimating absolute distances between camera centers of consecutive image frames. These estimates would improve the overall performance of classical (not deep) SLAM systems and allow metric feature locations to be recovered from a single monocular camera. We propose several network architectures that lead to an improvement of scale estimation accuracy over the state of the art. In addition, we exploit a possibility to train the neural network only with synthetic data derived from a computer graphics simulator. Our key insight is that, using only synthetic training inputs, we can achieve similar scale estimation accuracy as that obtained from real data. This fact indicates that fully annotated simulated data is a viable alternative to existing deep-learning-based SLAM systems trained on real (unlabeled) data. Our experiments with unsupervised domain adaptation also show that the difference in visual appearance between simulated and real data does not affect scale estimation results. Our method operates with low-resolution images (0.03MP), which makes it practical for real-time SLAM applications with a monocular camera.

Keywords: *machine learning, deep learning, SLAM, scale estimation*

For citation:

Rukhovich D. D. Estimation of Absolute Scale in Monocular SLAM Using Synthetic Data, *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 86—95

DOI: 10.17587/prin.11.86-95

References

1. **Shokurov A. V., Krivchikova K. A.** Absolute Error Estimation when Reconstructing the Extrinsic Parameters of a Camera Group in Application to Object Localization and Tracking, *Programmnaya Ingeneria,* 2017, vol. 8, no. 10, pp. 470—480 (in Russian).
2. **Tateno K., Tombari F., Laina I., Navab N.** CNN-SLAM: Real-time dense monocular slam with learned depth prediction, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017, pp. 6243—6252.
3. **Yang N., Wang R., Stuckler J., Cremers D.** Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry, *Proceedings of the European Conference on Computer Vision (ECCV),* 2018, pp. 817—833.
4. **Wang S., Clark R., Wen H.**, **Trigoni N.** Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2043—2050.
5. **Frost D., Murray D., Prisacariu V.** Using learning of speed to stabilize scale in monocular localization and mapping, *2017 International Conference on 3D Vision (3DV)*, IEEE, 2017, pp. 527—536.

6. **Geiger A., Lenz P., Urtasun R.** Are we ready for autonomous driving? The KITTI vision benchmark suite, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3354—3361.
7. **Scaramuzza D., Fraundorfer F., Pollefeys M., Siegwart R.** Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic *constraints, 2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 1413—1419.
8. **Atapour-Abarghouei A., Breckon T. P.** Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2800—2810.
9. **Dosovitskiy A., Fischer P., Ilg E.** et al. Flownet: Learning optical flow with convolutional networks, *Proceedings of the IEEE International conference on computer vision*, 2015, pp. 2758—2766.
10. **Rosenholm D. A. N., Torlegar D. K.** Three-dimensional absolute orientation of stereo models using digital elevation models, *Photogrammetric engineering and remote sensing*, 1988, vol. 54, no. 10, pp. 1385—1389.
11. **Geiger A., Lauer M., Wojek C.** et al. 3d traffic scene understanding from movable platforms, *IEEE transactions on pattern analysis and machine intelligence*, 2013, vol. 36, no. 5, pp. 1012—1025.
12. **Geiger A., Ziegler J., Stiller C.** Stereoscan: Dense 3d reconstruction in real-time, *2011 IEEE intelligent vehicles symposium (IV),* IEEE, 2011, pp. 963—968.
13. **Gräter J., Schwarze T., Lauer M.** Robust scale estimation for monocular visual odometry using structure from motion and vanishing points, *2015 IEEE Intelligent Vehicles Symposium (IV),* IEEE, 2015, pp. 475—480.
14. **Zhou D., Dai Y., Li H.** Ground-Plane-Based Absolute Scale Estimation for Monocular Visual Odometry, *IEEE Transactions on Intelligent Transportation Systems*, 2019, vol. 21, pp. 791—802.
15. **Castle R., Gawley D., Klein G, Murray D. W.** Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras, *Proceedings 2007 IEEE International Conference on Robotics and Automation,* IEEE, 2007, pp. 4102—4107.
16. **Frost D., Kähler O., Murray D.** Object-aware bundle adjustment for correcting monocular scale drift, *2016 IEEE International Conference on Robotics and Automation (ICRA),* IEEE, 2016, pp. 4770—4776.
17. **Kitt B., Geiger A., Lategahn H.** Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme, *2010 IEEE intelligent vehicles symposium,* IEEE, 2010, pp. 486—492.
18. **Song S., Chandraker M.** Robust scale estimation in real-time monocular SFM for autonomous driving, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2014, pp. 1566—1573.
19. **Sucar E., Hayet J. B.** Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift, *2018 IEEE International Conference on Robotics and Automation (ICRA),* IEEE, 2018, pp. 1—7.
20. **Izadi S., Kim D., Hilliges O.** et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera, *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559—568.
21. **Kerl C., Sturm J., Cremers D.** Robust odometry estimation for RGB-D cameras, *2013 IEEE international conference on robotics and automation,* IEEE, 2013, pp. 3748—3754.
22. **Eigen D., Puhrsch C., Fergus R.** Depth map prediction from a single image using a multi-scale deep network, *Advances in neural information processing systems,* 2014, pp. 2366—2374.
23. **Fu H., Gong M., Wang C.** et al. Deep ordinal regression network for monocular depth estimation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002—2011.
24. **Hoffman J., Tzeng E., Park T.** et al. Cycada: Cycle-consistent adversarial domain adaptation, arXiv preprint arXiv:1711.03213. 2017.
25. **Zheng C., Cham T., Cai J.** T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks, *Proceedings of the European Conference on Computer Vision (ECCV),* 2018, pp. 767—783.
26. **Dosovitskiy A., Ros G., Codevilla F.** et al. CARLA: An open urban driving simulator, arXiv preprint arXiv:1711.03938. 2017.
27. **Quiter C., Ernst M.** deepdrive/deepdrive: 2.0. 2018, available at: https://github.com/deepdrive/deepdrive
28. **Shah S., Dey D., Lovett C., Kapoor A.** Airsim: High-fidelity visual and physical simulation for autonomous vehicles, *Field and service robotics*, Springer, Cham, 2018, pp. 621—635.
29. **Goodfellow I., Bengio Y., Courville A.** Deep learning, MIT press, 2016, 776 p.
30. **Hochreiter S., Schmidhuber J.** Long short-term memory, *Neural computation,* 1997, vol. 9, no. 8, pp. 1735—1780.
31. **Donahue J., Hendricks L. A., Rohrbach M.** et al. Long-term recurrent convolutional networks for visual recognition and description, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625—2634.
32. **Abadi M., Agarwal A., Barham P.** et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467. 2016.
33. **Kingma D. P., Ba J.** Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980. 2014.
34. **Mayer N., Ilg E., Fischer P.** et al. What makes good synthetic training data for learning disparity and optical flow estimation? *International Journal of Computer Vision*, 2018, vol. 126, no. 9, pp. 942—960.

ИНФОРМАЦИЯ

3-я Международная научно-техническая конференция

"СОВРЕМЕННЫЕ СЕТЕВЫЕ ТЕХНОЛОГИИ"

Modern Network Technologies, MoNeTec-2020

27—29 октября 2020 г., Москва

Конференция собирает представителей международного научного сообщества, исследовательских подразделений корпораций, стартапов, промышленности и бизнеса, институтов развития и органов государственной власти для обсуждения перспективных и актуальных технологий в сфере компьютерных сетей, виртуализации сетевых ресурсов и облачных вычислений, использования методов искусственного интеллекта.

Подробная информация по условиям участия на сайте конференции
http://www.monetec.ru

УДК 519.873 DOI: 10.17587/prin.11.96-107

В. Н. Буков, д-р техн. наук, гл. науч. сотр., v_bukov@mail.ru, ОАО "Бортовые аэронавигационные системы", г. Москва, **А. М. Агеев,** канд. техн. наук, нач. отдела — зам. начальника управления, ageev_bbc@mail.ru, **А. М. Мальцев,** канд. техн. наук, ст. науч. сотр., max_alex_67@mail.ru, Военный учебно-научный центр Военно-воздушных сил "Военно-воздушная академия им. проф. Н. Е. Жуковского и Ю. А. Гагарина", г. Воронеж

Программный инструментарий поддержки синтеза системы управления избыточностью комплекса бортового оборудования

Рассмотрены компьютерные средства разработки комплексов бортового оборудования технических систем с управляемой избыточностью. Эти средства охватывают такие фазы, как задание исходных данных, включая номинальную конфигурацию, генерацию альтернативных конфигураций и формирование таблицы конфигураций избыточного комплекса бортового оборудования. Приведен методический пример.

Ключевые слова: *интегрированный комплекс бортового оборудования, интегрированная модульная авионика, управление избыточностью, генерация альтернативных конфигураций, супервизоры конфигураций*

Введение

Получившая поддержку в официальных нормативных документах [1] концепция интегрированной модульной авионики (ИМА) заняла одно из центральных мест в работах зарубежных [2—5] и отечественных [6—8] специалистов по авиационному оборудованию и стала основой для разработки современных бортовых комплексов. Новые разработки нацелены на придание комплексам бортового оборудования (КБО), основанным на ИМА, значительно улучшенных эксплуатационных свойств.

К направлениям развития концепции ИМА относится и разработка авионики необслуживаемого бортового оборудования (АНБО). Такое оборудование предусматривает [8] существенное увеличение интервалов времени между процедурами его обслуживания за счет избыточных бортовых ресурсов (вычислители, каналы передачи данных, периферийные системы, источники данных и исполнительные устройства).

Вместе с тем как теоретической, так и практической основой создания АНБО является интенсивно развиваемая технология управления избыточностью комплексов оборудования. Используемый здесь термин "избыточность" подразумевает, что число входящих в КБО компонентов заведомо превышает минимально необходимый уровень, обусловленный предназначением комплекса. Таким образом, рассматривается прежде всего аппаратная избыточность, которая имеет в равной степени признаки структурной (резервирование) [9] и функциональной [10] избыточности. Такая избыточность может быть использована в интересах повышения общей производительности и обеспечения необходимого уровня безотказности в целях эффективного управления конфигурацией КБО. В статье не рассматриваются вопросы информационной и программно-логической избыточности, широко освещенные в работах [11, 12].

В основу реализации создаваемой технологии должны быть положены средства автоматизированного проектирования и поддержки жизненного цикла избыточных КБО. Такие средства должны активно использоваться в общем процессе разработки бортового радиоэлектронного оборудования перспективных авиационных комплексов. Общая архитектура инструментальных средств разработки избыточных КБО показана на рис. 1. Их основу составляют средства анализа, синтеза, представления и обработки информации о конфигурациях КБО, представленные в центральной части рисунка.

Настоящая статья посвящена разработке основ программных средств поддержки процессов создания и исследования характеристик КБО с управляемой избыточностью. К их числу относятся процедуры синтеза конфигураций и заполнения таблиц супервизоров конфигураций, выделенные на рис. 1 полужирной рамкой.

1. Состояние направления

В публикациях, посвященных развиваемой концепции управляемой избыточности в технических системах, последовательно излагаются вопросы:

- обоснования облика КБО с управляемой избыточностью [13];
- выбора критериев и показателей функциональной эффективности таких комплексов [14];

Рис. 1. Инструменты разработки избыточных КБО:

ВС — воздушное судно; ПО — программное обеспечение; БИВС — бортовая интегрированная вычислительная среда; СпПО — специальное программное обеспечение; СК — супервизор конфигурирования; ПФЭ — показатель функциональной эффективности

- формирования номинальной конфигурации комплекса с определением его целевой функции [15];
- генерирования альтернативных конфигураций комплекса, обеспечивающих неизменность его целевой функции [16];
- выбора предпочтительных для реализации конфигураций комплекса [14];
- составления таблиц супервизоров конфигураций комплекса [17];
- организации БИВС для реализации управления избыточностью [18] и ряд смежных вопросов.

Весь комплекс исследований, включая настоящую работу, нацелен на создание эффективного подхода к синтезу сложных многосвязных (с десятками компонентов) систем с развитыми средствами компьютерной поддержки. Вместе с тем пока успехи ограничены тестовыми примерами с числом компонентов в пределах одного десятка.

2. Общее описание избыточной системы

В основе технологии избыточности лежит понятие конфигурации КБО или в расширенном толковании — конфигурации КО (комплекса оборудования). Предметом автоматического и/или автоматизированного выбора (изменения, исследования) конфигураций являются технические (аппаратные и/или программные) средства управления избыточностью системы, структура которых показана на рис. 2.

Элементы структуры связаны между собой в соответствии с линеаризованными уравнениями функционирования объекта управления совместно с избыточными компонентами КБО:

$$y_\tau = Dx_\tau,\ x_{\tau+1} = Ax_\tau + Bu_\tau + Gv_\tau,\ x_{\tau=0} = x_0 \qquad (1)$$

и бортовой интегрированной вычислительной среды (БИВС):

$$u_\tau = Q(z)y_\tau = C_{\text{вх}}E(z)C_{\text{вых}}y_\tau. \qquad (2)$$

В представленных выражениях: y_τ — метавектор (составной вектор) выходов всех компонентов на такте τ размерности m; x_τ — метавектор состояния компонентов и объекта размерности n; u_τ — метавектор входов компонентов для межкомпонентных связей размерности l; v_τ — метавектор входов компонентов и объекта для внешних воздействий (внешних входов) размерности k; D — числовая матрица формирования выходов всех компонентов размеров $m \times n$; A — числовая матрица собственной динамики компонентов и объекта размеров $n \times n$; B — числовая матрица эффективности межкомпонентных связей размеров $n \times l$; G — числовая матрица эффективности внешних воздействий размеров $n \times k$; z — оператор сдвига во времени на один такт вперед; $Q(z)$ — дробно-рациональная полиномиальная (по оператору z) матрица размеров $l \times m$ (передаточная матрица), названная конфигурационной матрицей [14]; $C_{\text{вх}}$ и $C_{\text{вых}}$ — рас-

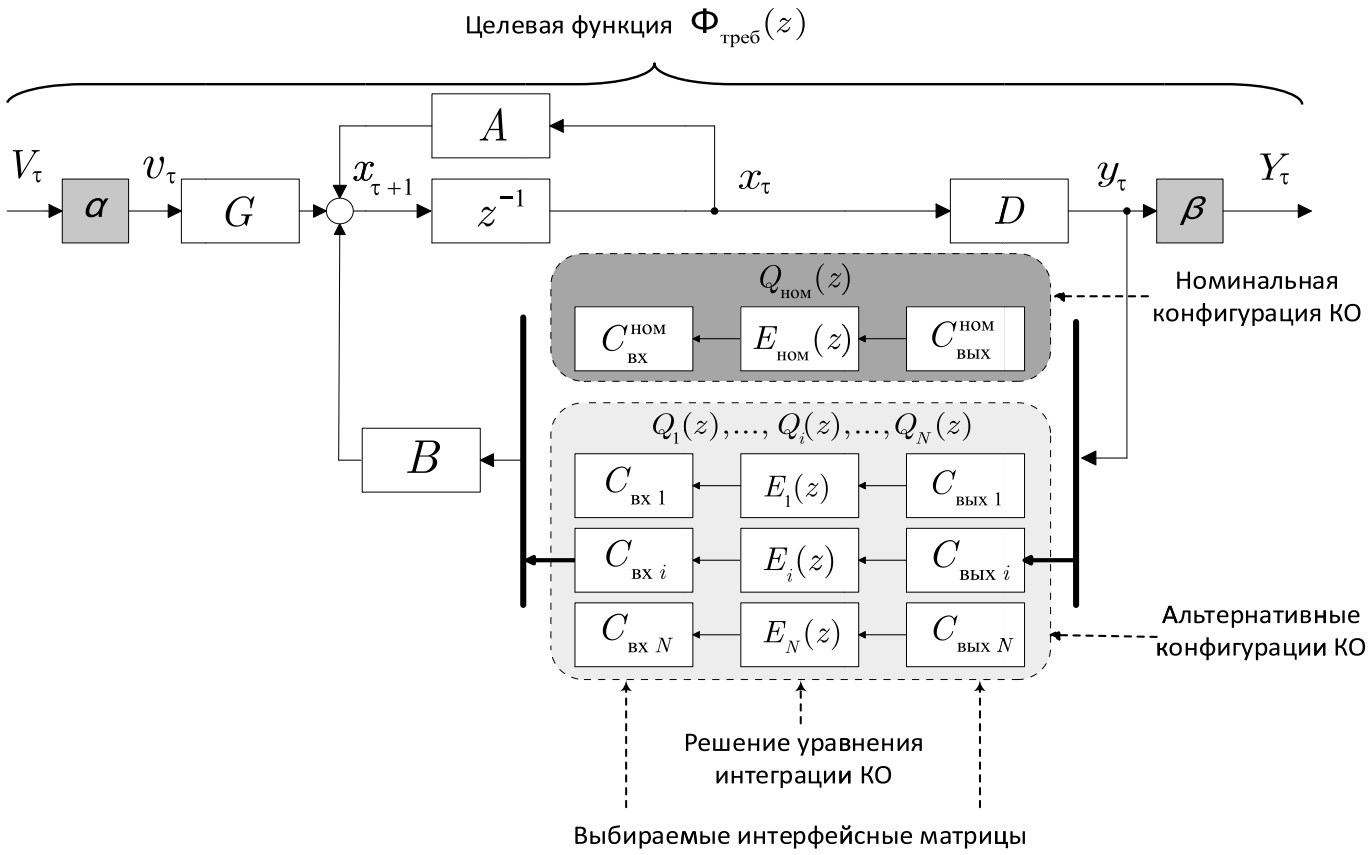

Рис. 2. Структура системы "Объект + избыточный КБО"

пределительные матрицы размеров $l \times p$ и $q \times m$, т. е. матрицы, содержащие бинарные элементы и не более одного единичного элемента в строке, названные интерфейсными (входной и выходной по отношению к компонентам КО) матрицами [16]; $E(z)$ — дробно-рациональная полиномиальная матрица размеров $p \times q$, моделирующая обработку вычислительными средствами БИВС всех поступающих данных и названная интеграционной матрицей [15].

При этом передаточная матрица от "тестового" входа V_τ к "тестовому" выходу Y_τ определяется как

$$\Phi(z) = \beta\left[w_{y_j}^{v_i}(z) \right]_{m \times k} \alpha =$$

$$= \beta D\left(zI_n - A - BC_{\text{вх}}E(z)C_{\text{вых}}D\right)^{-1} G\alpha.$$

Здесь α и β — весовые матрицы размеров $k \times g$ и $f \times m$; $w_{y_j}^{v_i}(z)$ — передаточная функция по внешнему воздействию от i-го входа $v_{i.\tau}$ к j-му выходу $y_{j.\tau}$; $m \times k$ — размеры передаточной матрицы $W_y^v(z) = \left[w_{y_j}^{v_i}(z) \right]$; I_n — единичная матрица размеров $n \times n$. Функция $\Phi(z)$ именуется целевой функцией [15] и по определению отражает содержание и качество выполнения системой "Объект + избыточный КБО" надлежащих функций.

Одна из конфигураций КБО (подразумеваются фиксированные числовые значения интерфейсных $C_{\text{вх}}^{\text{ном}}$, $C_{\text{вых}}^{\text{ном}}$ и дробно-рациональное значение интеграционной $E_{\text{ном}}(z)$ матриц), полученная каким-либо способом (по прототипу, традиционными инженерными способами, синтезом на основе оптимизационных и других подходов) и удовлетворяющая разработчика системы, называется номинальной [15].

В соответствии с концепцией управления избыточностью к альтернативным конфигурациям избыточного КБО относятся все конфигурации $C_{\text{вх}}$, $C_{\text{вых}}$ и $E(z)$, отличные от номинальной $C_{\text{вх}}^{\text{ном}}$, $C_{\text{вых}}^{\text{ном}}$ и $E_{\text{ном}}(z)$, но обеспечивающие для системы (1), (2) неизменность соответствующего ей значения целевой функции

$$\Phi_{\text{треб}}(z) = \beta D\left(zI_n - A - BC_{\text{вх}}^{\text{ном}}E_{\text{ном}}(z)C_{\text{вых}}^{\text{ном}}D\right)^{-1} G\alpha. \quad (3)$$

Процесс получения альтернативных конфигураций, составляющий значительную часть разработанного инструментария, содержит выполнение следующих процедур [13]:

- тестирования предварительно сформированных конфигураций на допустимость проверкой равенств для матриц $C_{\text{вх}}$ и $C_{\text{вых}}$:

$$\overline{BC_{\text{вх}}}^{L} Q_{\text{ном}}(z)D = 0 \ \text{ и } \ BQ_{\text{ном}}(z)\overline{C_{\text{вых}}D}^{R} = 0; \quad (4)$$

- интеграции КБО по формуле для множества решений основного уравнения

Таблица 1

Вариант представления супервизоров конфигураций

Порядковые номера СК	Порядковые номера компонентов					ИГ СК	ПФЭ
	1	2	3	...	K		
	Наименования и ИГ компонентов						
	K_1	K_2	K_3	...	K_K		
	1	1	0	...	1		
	Порядковые номера компонентов $\frac{\text{на входе}}{\text{на выходе}}$						
1	4	1	1	...	2, 3	1	E_1
	2, 3	4	4	...	1		
2	4	1	0	...	0	0	E_2
	2	K-1	0	...	0		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
N	0	1	0	...	K-2	1	E_N
	0	4	0	...	1		

$$\underbrace{\{E(z)\}_{\theta,\vartheta}}_{\text{Общее решение}} = \underbrace{\left(\beta D\Omega^{-1}(z)BC_{\text{вх}}\right)^{\sim}\overbrace{\beta D\Omega^{-1}(z)BQ_{\text{ном}}(z)D\Omega^{-1}(z)G\alpha}^{\text{Интегрирующий базис}}\left(C_{\text{вых}}D\Omega^{-1}(z)G\alpha\right)^{\sim}}_{\text{Базовое решение}} + \tag{5}$$

$$+ \underbrace{\overline{\beta D\Omega^{-1}(z)BC_{\text{вх}}}^{R}\theta}_{\text{Вариация столбцов решения}} + \underbrace{\vartheta\overline{C_{\text{вых}}D\Omega^{-1}(z)G\alpha}^{L}}_{\text{Вариация строк решения}},$$

где $\Omega(z) = zI_n - A - BQ_{\text{ном}}(z)D$; θ и ϑ — произвольные матрицы подходящих размеров.

Указанные здесь операции канонизации произвольной матрицы M (вычисление канонизатора $(M)^{\sim}$, а также левого $\overline{M}^{L}$ и правого $\overline{M}^{R}$ делителей нуля максимального ранга [13]) не относятся к широко распространенным и потребовали специальных усилий при программировании.

3. Формулировка задачи и супервизоры конфигураций

При использовании представленной математической модели аналитический синтез избыточного КБО сводится к поиску, во-первых, номинальной конфигурации $Q_{\text{ном}}(z)$ и по формуле (3) требуемой целевой функции $\Phi_{\text{треб}}(z)$ на основе исходных данных о системе (1), а во-вторых, по формулам (4) и (5) множества альтернативных конфигураций $Q(z)$, обеспечивающих неизменность целевой функции (3). Целью настоящей статьи является представление средств компьютеризации этих процессов вместе с последующим процессом формирования так называемых супервизоров конфигураций [17], предназначенных для хранения и реализации сгенерированных конфигураций КБО.

Под такими супервизорами понимаются аппаратные и/или программные модули, предназначенные для мониторинга работоспособности своей конфигурации, для использования в межсупервизорном арбитраже, коммутации и инициировании своей конфигурации при победе в арбитраже [13].

Супервизоры конфигураций могут быть представлены в виде, иллюстрируемом табл. 1. В этой таблице N — число конфигураций КБО; K — число компонентов; ИГ — индексы готовности компонентов и конфигураций (супервизоров); ПФЭ — показатели функциональной эффективности (по совокупности признаков) конфигураций[1].

Таким образом, для каждой альтернативной конфигурации полученные матрицы $C_{\text{вх}}$ и $C_{\text{вых}}$ используются для формирования супервизора конфигурации, управляющего коммуникационными устройствами КБО, а матрица $E(z)$ — для программирования вычислительных средств БИВС, реализующих конфигурацию.

[1] Вопросы, связанные с мониторингом работоспособности (формированием ИГ) и оценкой эффективности (формированием ПФЭ), в данной работе не рассматриваются.

Реализуя поставленную задачу, авторы использовали[2] высокоуровневый язык и интерактивную среду программирования, численных расчетов и визуализации результатов MATLAB со специальными инструментальными средствами визуально-ориентированного программирования и проектирования приложений с графическим интерфейсом пользователя *GUI*. Язык и средства программирования включают в себя такие средства, как конструктор графического интерфейса *GUIDE*, инспектор свойств *Property Inspector*, функции обработки событий для компонентов и окон *Callbacks*, редактор *M*-файлов *M-file Editor* и т. д. [19, 20]. Выбор таких инструментальных средств обоснован ориентированностью механизмов MATLAB к матричным операциям. В качестве математической основы технологии управления избыточностью выступают методы решения матричных уравнений с использованием канонизации матриц [15]. Использование этих средств обусловлено также возможностью последующей адаптации алгоритмов в программный код, подобный коду, получаемому на языке программирования C++.

4. Общая структура программы

Алгоритм представляемой вычислительной программы показан на рис. 3.

Как следует из рис. 3, алгоритм состоит из двух частей. Первая часть отвечает за ввод исходных данных (функция *Super*_1.*m*), а вторая часть — за непосредственное решение задачи (функция *Sintez*_2.*m*), а именно:

- определение номинальной конфигурационной матрицы и требуемой целевой функции;
- выбор и тестирование на допустимость по формулам (4) пары интерфейсных матриц $C_{вх}$ и $C_{вых}$, при которых возможен учет дополнительных требований разработчика системы (предпочтительный состав используемых компонентов и пр.);
- определение по формуле (5) с использованием аппарата канонизации матриц (подфункция *Canoniz.m*) интеграционной матрицы E для выбранной пары $C_{вх}$ и $C_{вых}$, когда возможен учет дополнительных требований разработчика системы (предпочтительный состав вычислительных процедур в БИВС);
- проверку выполнения заданных требований (3) (неизменность заданной целевой функции);
- формирование супервизора конфигурации ($SK_{вх}$, $SK_{вых}$) по тройке матриц ($C_{вх}$, $C_{вых}$, $E(z)$) для допустимой конфигурации с учетом критериев разработчика;
- запоминание и вывод полученных результатов в удобном для пользователя виде.

5. Алгоритм программы генерации конфигураций

Алгоритм реализует процессы, отмеченные в разд. 3.

Блок выбора интерфейсных матриц $C_{вх}$ и $C_{вых}$ позволяет: задать их в виде номинальных значений $C_{вх}^{ном}$, $C_{вых}^{ном}$ (например, для тестирования программы); изменять вручную элементы каждой из них; зафиксировать одну из них и сгенерировать все возможные варианты другой, сгенерировать возможные варианты обеих матриц одновременно.

При генерации вариантов пар интерфейсных матриц $C_{вх}$ и $C_{вых}$ применяются некоторые, включая не фигурирующие явно в теоретических положениях, допущения и ограничения, для пояснения которых используется рис. 4.

Для такой структуры КБО основные положения сводятся к перечисленным далее:

а) размеры всех матриц $C_{вх}$, $C_{вых}$, $E(z)$ известны и зафиксированы;

б) в матрице $E(z)$:

1) число строк p равно числу каналов вычислений, которое с учетом резервирования должно быть не меньше числа устройств, подключенных к входу/выходу вычислительной системы (B_1, B_2, …, B_p), т. е. $p \geqslant \max(l, m)$, где l — число исполнительных устройств u_τ, m — число датчиков y_τ;

2) число столбцов q равно числу вариантов передаточных функций (ПФ) и с учетом их вариации должно быть не меньше числа датчиков y_τ, т. е. $p \geqslant m$;

в) в матрице $C_{вых}$:

1) число строк q равно числу вариантов ПФ (см. п. 2 для матрицы E);

2) число столбцов равно числу датчиков m;

3) в строке не должно быть больше одной "1" (на каждый вход вычислителя не может быть подано более одного сигнала из всей совокупности датчиков);

4) значение "1" в элементе $C_{вых.i,j}$ означает, что сигнал j-го датчика поступает на i-й вход вычислителя (умножается на i-й вариант ПФ в соответствующем вычислителе).

г) в матрице $C_{вх}$:

1) число строк l равно числу исполнительных устройств (актюаторов или индикаторов);

2) число столбцов равно числу каналов вычислений p;

3) в строке не должно быть больше одной "1" (на каждое исполнительное устройство нельзя подать больше одного сигнала из всей совокупности каналов вычислений);

4) символ "1" в элементе $C_{вх.i,j}$ означает, что к i-му исполнительному устройству подключается j-й вычислительный канал.

Результатом работы алгоритма генерации конфигурации является выбор альтернативных конфигураций комплекса, обеспечивающих неизменность его целевой функции и формирование таблицы супервизоров конфигураций КБО.

6. Алгоритм программы формирования супервизоров конфигураций

Для найденных номинальной и альтернативных конфигураций формируется таблица супервизоров конфигураций комплекса. Здесь принципиальным является то, что в результате должны определиться не все потенциально возможные связи компонентов системы, а только те, которые при соответствующих

[2] Лицензия № 989813.

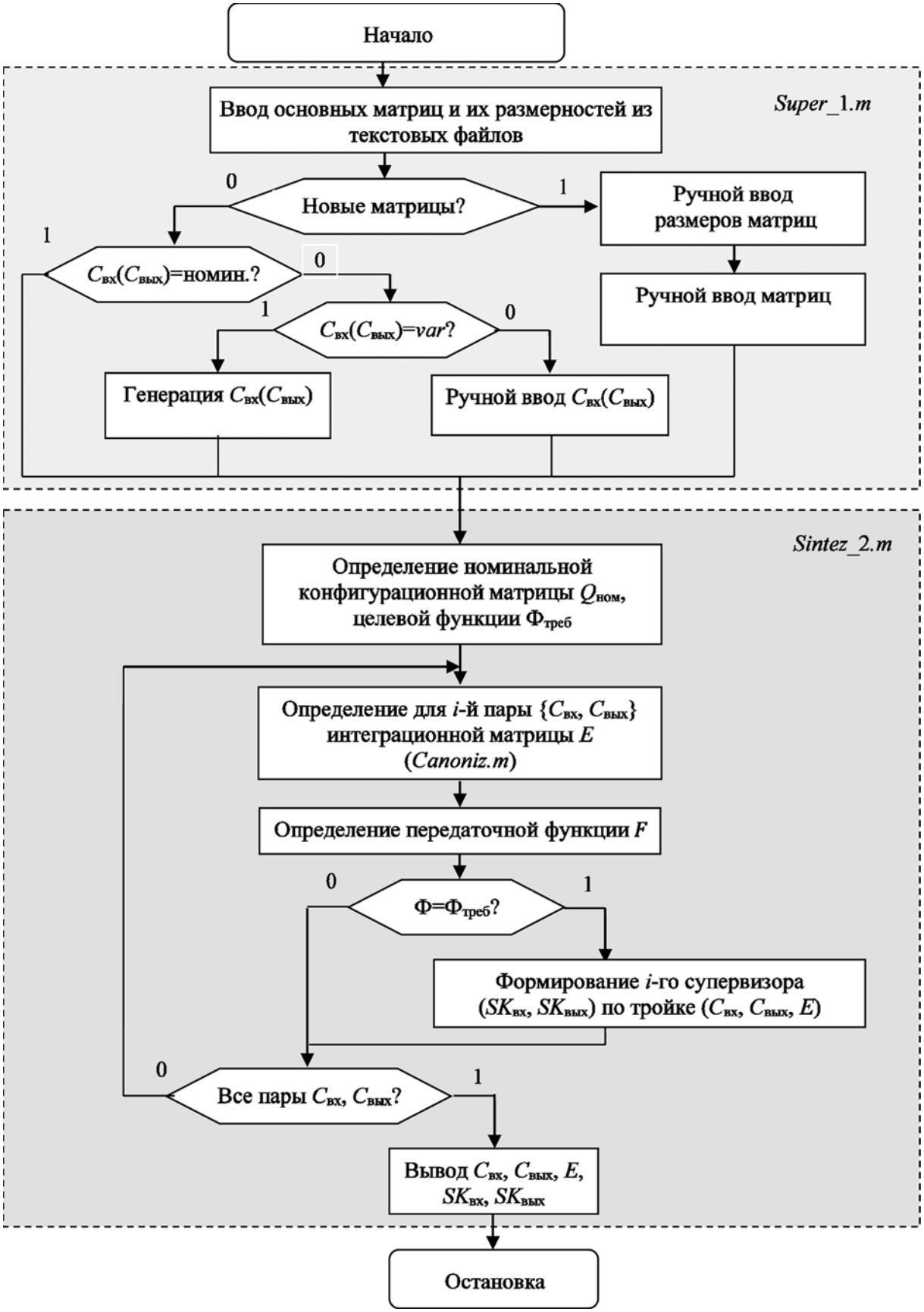

Рис. 3. Алгоритм решения задачи синтеза избыточных КБО

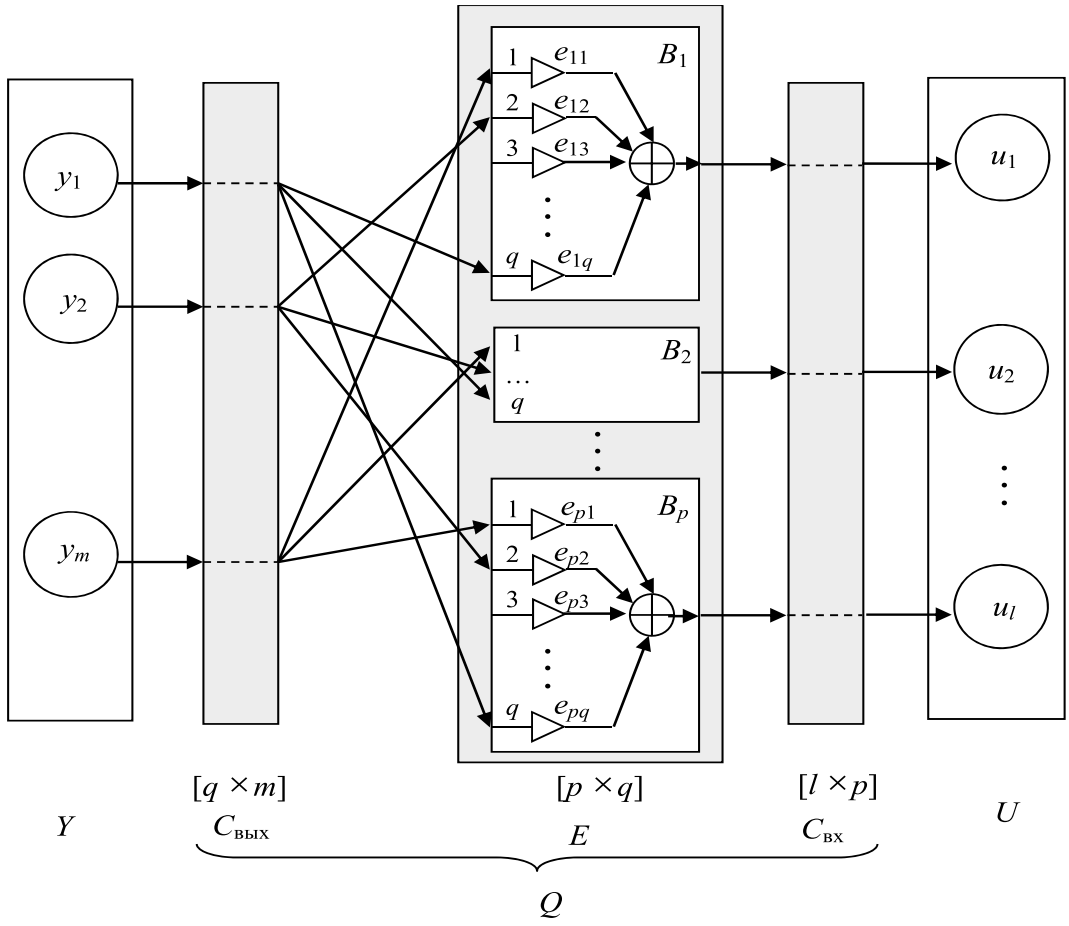

Рис. 4. Унифицированная структура КБО, используемая программой

настройках вычислителей могут обеспечить неизменность целевой функции $\Phi_{треб}(z)$.

Соответствующий алгоритм заключается в следующем.

Шаг 1. Формируется пустая таблица СК комплекса всех компонентов.

Шаг 2. Для каждой конфигурации, т. е. строки таблицы.

Шаг 2.1. Формируется матрица V размеров матрицы $E(z)$ с номерами строк в качестве элементов, например

$$V = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}.$$

Шаг 2.2. Каждый элемент матрицы V^* приравнивается нулю, если одноименный элемент матрицы E равен нулю, например,

$$\text{если } E = \begin{bmatrix} 0 & e_{12} & 0 \\ e_{21} & 0 & e_{23} \\ 0 & 0 & e_{33} \end{bmatrix}, \text{ то } V^* = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 3 \end{bmatrix}.$$

Шаг 2.3. Анализируется произведение матриц $SK = C_{вх}V^*C_{вых}$:

- номер столбца определяет номер датчика, а ненулевые элементы столбца — номер соответствующего вычислительного канала;
- номер строки определяет номер исполнительного устройства, а ненулевые элементы строки — номер выхода вычислительного канала.

Например, третий столбец матрицы

$$C_{вх}V^*C_{вых} = \begin{array}{c} \begin{matrix} Д_1 & Д_2 & Д_3 & Д_4 & Д_5 \end{matrix} \\ \begin{bmatrix} 0 & 5 & \boxed{0} & 0 & 0 \\ 0 & 0 & \boxed{2} & 0 & 0 \\ 3 & 0 & \boxed{4} & 0 & 0 \\ 0 & 0 & \boxed{0} & \boxed{1} & 0 \end{bmatrix} \end{array} \begin{matrix} И_1 \\ И_2 \\ И_3 \\ И_4 \end{matrix}$$

↑ $В_1$ (номер вычислителя)

означает, что выход $Д_3$ подключен к входам вычислителя 2, выход которого, в свою очередь, подключен к входу исполнительного устройства $И_2$, и вычислителя 4, выход которого подключен к входу $И_3$;

Шаг 2.4. Заполняется строка таблицы в виде двух массивов $SK_{вх}$ и $SK_{вых}$, связанных с номерами устройств на входе и выходе устройств, номера которых соответствуют значениям ячейки.

Шаг 3. В результате анализа всех конфигураций формируется таблица СК комплекса.

7. Пример работы программы

Рассмотрим методический пример на основе линейной модели продольного движения самолета в приращениях относительно горизонтального полета с постоянной скоростью. Уравнение модели объекта имеет вид

$$\underbrace{\begin{bmatrix} x_{1.\tau+1} \\ x_{2.\tau+1} \\ x_{3.\tau+1} \end{bmatrix}}_{x_{\tau+1}} = \underbrace{\begin{bmatrix} a_1 & a_2 & 0 \\ a_3 & a_4 & 0 \\ 0 & a_2 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_{1.\tau} \\ x_{2.\tau} \\ x_{3.\tau} \end{bmatrix}}_{x_\tau} + \underbrace{\begin{bmatrix} 0 & 0 \\ b_1 & b_2 \\ 0 & 0 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} u_{1.\tau} \\ u_{2.\tau} \end{bmatrix}}_{u_\tau} + \underbrace{\begin{bmatrix} g_1 & 0 & g_2 \\ 0 & g_3 & g_4 \\ 0 & 0 & 0 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} v_{1.\tau} \\ v_{2.\tau} \\ v_{3.\tau} \end{bmatrix}}_{v_\tau}, \quad (6)$$

где x_1 — приращение угла атаки; x_2 — угловая скорость тангажа; x_3 — приращение угла тангажа; u_1 — приращение угла отклонения стабилизатора; u_2 — приращение угла отклонения переднего горизонтального оперения; v_1, v_2, v_3 — внешние воздействия (различные комбинации возмущений нормальной силы, продольного момента и сдвига ветра); a_i, b_i и g_i — известные параметры модели, отличные от нуля.

Выходом объекта является вектор y_τ, который определяется формулой

$$\underbrace{\begin{bmatrix} y_{1.\tau} \\ y_{2.\tau} \\ y_{3.\tau} \\ y_{4.\tau} \\ y_{5.\tau} \end{bmatrix}}_{y_\tau} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} x_{1.\tau} \\ x_{2.\tau} \\ x_{3.\tau} \end{bmatrix}}_{x_\tau} = \begin{bmatrix} x_{1.\tau} \\ x_{2.\tau} \\ x_{2.\tau} \\ x_{3.\tau} \\ x_{3.\tau} - x_{1.\tau} \end{bmatrix},$$

где y_1 — сигнал датчика угла атаки (ДУА); y_2 — сигнал датчика угловой скорости (ДУС) тангажа; y_3, y_4 и y_5 — сигналы угловой скорости тангажа; сигнал угла тангажа и угла наклона траектории, формируемый комплексной навигационной системой (КНС), соответственно.

Задачей управления является обеспечение желаемой передаточной матрицы от внешних воздействий v_1 и v_2 к угловой скорости тангажа $x_2 = y_2$, которая при заданных весовых матрицах

$$\alpha = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

определяется формулой

$$\Phi_{\text{треб}}(z) = \begin{bmatrix} 0 & \dfrac{g_3}{z - a^{\text{ж}}} \end{bmatrix}, \quad (7)$$

где $a^{\text{ж}}$ — параметр, характеризующий желаемые динамические свойства (переходные процессы) самолета.

Вариант конфигурации системы управления

$$Q_{\text{ном}} = \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{C_{\text{вх}}^{\text{ном}}} \underbrace{\begin{bmatrix} \dfrac{-a_3}{b_1} & \dfrac{a^{\text{ж}} - a_4}{b_1} & 0 \end{bmatrix}}_{E_{\text{ном}}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{C_{\text{вых}}^{\text{ном}}} =$$

$$= \begin{bmatrix} \dfrac{-a_3}{b_1} & \dfrac{a^{\text{ж}} - a_4}{b_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

принят в качестве номинального.

Требуется найти альтернативные конфигурации системы (3), обеспечивающие неизменность целевой функции (7).

Принято, что в примере генерируются и тестируются только варианты матрицы $C_{\text{вых}}$.

Графический интерфейс ввода/вывода исходных данных, образованный файлом *Super*_1.*fig* и *m*-файлом *Super*_1.*m* для решения поставленной задачи, представлен на рис. 5.

Интерфейс позволяет кроме автоматического режима перейти в ручной режим ввода значений матриц в численном и/или символьном виде, в том числе в виде полиномов, а также выводить значения матриц для визуализации и осуществлять при необходимости их непосредственную коррекцию.

В рассматриваемом примере необходимо провести автоматический ввод исходных данных кнопкой "Ввод" в поле 1, отмеченном на рис. 5, зафиксировать номинальный вариант матрицы входных интерфейсов, отметив позицию "$C_{\text{вых}} = C_{\text{вых}}^{\text{ном}}$" и зафиксировать ее кнопкой "Ввод" в поле 4, выделить позицию "Генерация $C_{\text{вых}}$" и зафиксировать ее кнопкой "Ввод" в поле 5, дать команду на решение задачи кнопкой "Решение задачи" в поле 6.

Фрагмент командного окна с протоколом выдачи полученного результата представлен на рис. 6. Здесь показаны: целевая функция $\Phi_{\text{треб}}$ (F) для номинальных $C_{\text{вх}}^{\text{ном}}$, $C_{\text{вых}}^{\text{ном}}$; вычисленная интегрированная матрица E для выбранной пары $C_{\text{вх}}$, $C_{\text{вых}}$; число используемых объектом информационных, вычислительных и исполнительных элементов; супервизор конфигурации в виде двух массивов $SK_{\text{вх}}$ и $SK_{\text{вых}}$.

Выполненными расчетами были получены 58 вариантов конфигураций комплекса. Выборочные результаты представлены в табл. 2.

В табл. 2 сведены варианты с использованием сигналов:

1) ДУА и угловой скорости тангажа КНС;
2) ДУА и угла тангажа КНС;
3) ДУА, ДУС и угловой скорости тангажа КНС;
4) ДУА, угла тангажа и угловой скорости тангажа КНС.

Полученные супервизоры для номинальной, а также рассчитанных альтернативных конфигураций представлены в табл. 3. Здесь для номинальной (вариант 0) конфигурации КБО сигналы с датчиков $Д_1$ и $Д_2$ подаются на вход вычислителя $В_1$, в котором они преобразовываются в управляющий сигнал для $И_1$.

Аналогично проводится чтение табл. 3 для других вариантов.

Рис. 5. Графический интерфейс ввода/вывода данных в среде *GUIDO*:

1 — блок автоматического ввода матриц компонентов объекта; *2* — блок ввода размерностей компонентов объекта, интеграционной и целевой матриц; *3* — блок ручного ввода матриц компонентов объекта; *4* — блок выбора матрицы $C_{вх}$; *5* — блок выбора матрицы $C_{вых}$; *6* — пуск решения задачи

Таблица 2

Результаты работы программы

№ варианта	Интерфейсная матрица $C_{вых}$	Интеграционная матрица E
0 (ном)	$\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}\frac{-a_3}{b_1} & \frac{a_ж - a_4}{b_1} & 0\end{bmatrix}$
1	$\begin{bmatrix}1&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}\frac{-a_3}{b_1} & \frac{a_ж - a_4}{b_1} & 0\end{bmatrix}$
2	$\begin{bmatrix}1&0&0&0&0\\0&0&0&1&0\\0&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}\frac{-a_3}{b_1} & \frac{(a_ж - a_4)(z-1)}{a_2 b_1} & 0\end{bmatrix}$
3	$\begin{bmatrix}0&1&0&0&0\\1&0&0&0&0\\0&0&1&0&0\end{bmatrix}$	$\begin{bmatrix}\frac{a_ж - a_4}{b_1} & \frac{-a_3}{b_1} & 0\end{bmatrix}, \begin{bmatrix}0 & \frac{-a_3}{b_1} & \frac{a_ж - a_4}{b_1}\end{bmatrix}, \begin{bmatrix}\frac{a_ж - a_4}{2b_1} & \frac{-a_3}{b_1} & \frac{a_ж - a_4}{2b_1}\end{bmatrix}$
4	$\begin{bmatrix}0&0&0&1&0\\1&0&0&0&0\\0&0&1&0&0\end{bmatrix}$	$\begin{bmatrix}0 & \frac{-a_3}{b_1} & \frac{(a_ж - a_4)(z-1) - a_2 a_3}{b_1(z-1)}\end{bmatrix}, \begin{bmatrix}\frac{(a_ж - a_4)(z-1) - a_2 a_3}{a_2 b_1} & \frac{-a_3}{b_1} & 0\end{bmatrix}$

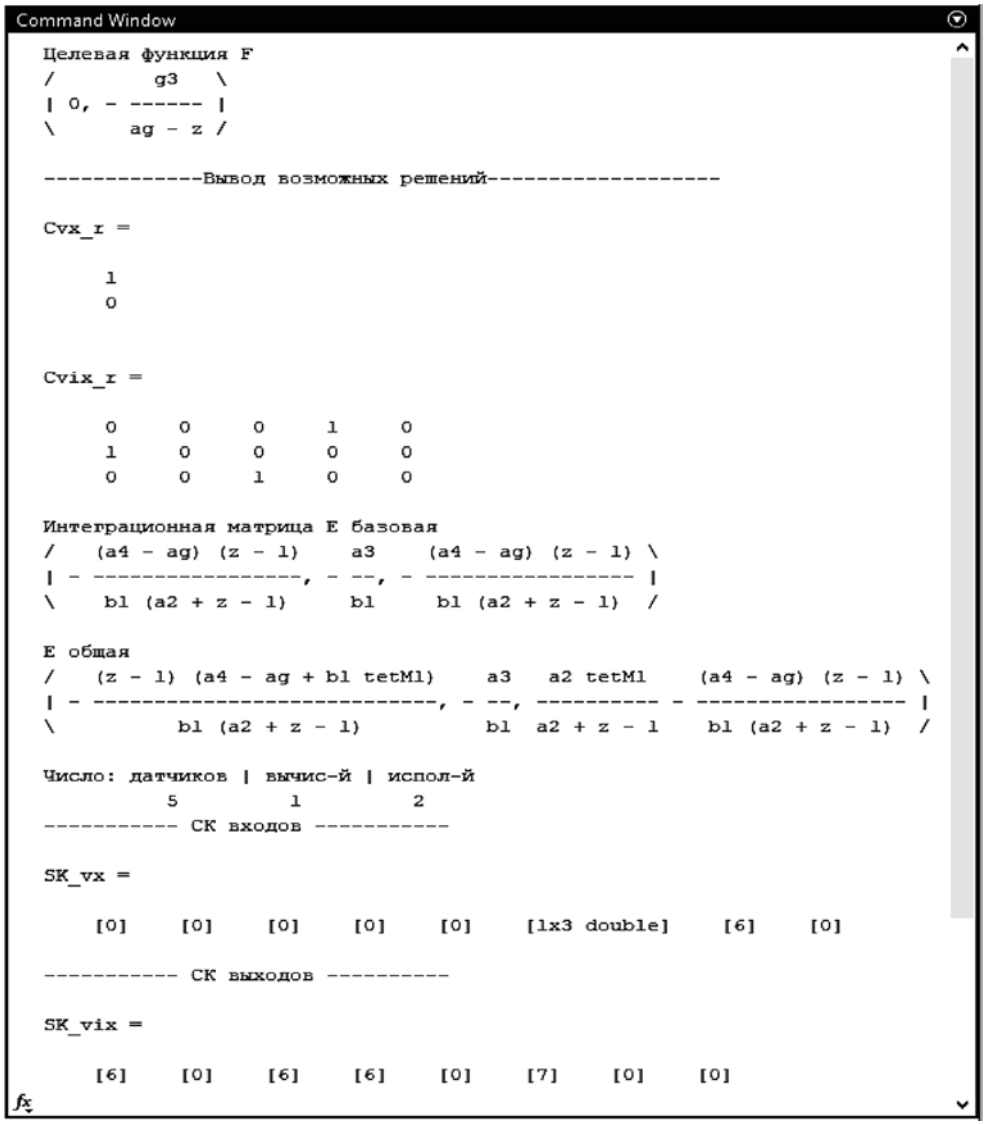

Рис. 6. Окно вывода результата работы программы

Таблица 3

Сформированные супервизоры конфигураций

№ СК	Порядковые номера компонентов							
	1	2	3	4	5	6	7	8
	Наименования компонентов							
	$Д_1$	$Д_2$	$Д_3$	$Д_4$	$Д_5$	$В_1$	$И_1$	$И_2$
	Порядковые номера компонентов: на входе / на выходе							
0 (ном)						1, 2	6	
	6	6				7		
1						1, 3	6	
	6		6			7		
2						1, 4	6	
	6			6		7		
3						1, 2, 3	6	
	6	6	6			7		
4						1, 3, 4	6	
	6		6	6		6		

Заключение

Разработанная программа выполняет генерацию альтернативных конфигураций избыточного многокомпонентного комплекса оборудования, обеспечивающих выполнение требуемой целевой функции, и формирование супервизоров конфигураций, реализующих такие конфигурации.

Дальнейшие исследования будут продолжены в направлении ранжирования конфигураций избыточного многокомпонентного объекта, обеспечивающих выполнение требуемой целевой функции по показателям эффективности. Это позволит организовать выбор предпочтительных конфигураций как на этапе проектирования КБО, так и в процессе его функционирования в целях парирования последствий деградации компонентов.

Список литературы

1. **DO-297.** Integrated modular avionics (IMA) development guidance and certification considerations. Washington: RTCA Inc., 2005.
2. **Watkins C. B.** Integrated Modular Avionics: Managing the Allocation of Shared Intersystem Resources // 25th Digital Avionics Systems Conference (DASC), 15-19 October 2006. Portland, Oregon, USA. URL: https://ieeexplore.ieee.org/document/4106349
3. **Garside R., Pighetti J. F.** Integrating Modular Avionics: A New Role Emerges // 26th Digital Avionics Systems Conference (DASC). 21-25 October 2007. Dallas, Texas, USA. URL: https://ieeexplore.ieee.org/document/4391843
4. **Hainaut D.** Towards the Next Generation of Integrated Modular Avionics // Aerodays. 30 March — 01 April 2011. Madrid, Spain. URL: http://www.pplane-project.org/Download/News/Info/document/1007.pdf
5. **Bernard S., Garcia J.-P.** Braking Systems with New IMA Generation // Aerospace Technology Conference and Exposition. SAE International, October 2011. Paper 2011-01-2662. DOI: 10.4271/2011-01-2662.
6. **Буздалов Д. В., Зеленов С. В., Корныхин Е. В.** и др. Инструментальные средства проектирования систем интегрированной модульной авионики // Труды Института системного программирования РАН. 2014. Т. 26, Вып. 1. С. 201—230.
7. **Дегтярев А. Р., Медведев Г. В.** Алгоритм распределения задач в многопроцессорных комплексах интегрированной модульной авионики // Автоматизация процессов управления. 2014. № 1 (35). С. 79—84.
8. **Bukov V., Kutahov V., Bekkiev A.** Avionics of Zero Maintenance Equipment // 27th Cong. of the Int. Council of the Aeronaut. Sciences 19—24 September. 2010, Nice, France. 2010. Paper 7-1-1.
9. **Аверьянов А.В., Барановский А. М., Эсаулов К. А.** Определение пределов аппаратной избыточности управляющих систем// Известия высших учебных заведений. Приборостроение. 2014. Т. 57, № 3. С. 23—26.
10. **Шульга Т. Э.** Общая схема управления дискретными системами на основе функциональной избыточности // Современные технологии. Системный анализ. Моделирование. 2010. № 1. С. 167—174.
11. **Васенин В. А., Пирогов М. В., Чечкин А. В.** Основной оператор радикального моделирования на демонстрационном примере // Программная инженерия. 2016. Т. 7, № 2. С. 75—85.
12. **Руднев Ю. П., Хетагуров Я. А.** Повышение надежности цифровых устройств методами избыточного кодирования. М.: Энергия, 1974. 272 с.
13. **Буков В. Н., Бронников А. М., Агеев А. М.** и др. Концепция управляемой избыточности комплексов бортового оборудования // Науч. чтения по авиации, посв. пам. Н. Е. Жуковского: Матер. XVI Всерос. науч.-практ. конф. / Гл. ред. С. П. Халютин. М.: Изд. дом Акад. им. Н. Е. Жуковского, 2019. С. 17—33.
14. **Агеев А. М.** Конфигурирование избыточных комплексов бортового оборудования // Изв. РАН. Теория и системы управления. 2018. № 4. С. 175—192.
15. **Буков В. Н., Бронников А. М., Агеев А. М., Гамаюнов И. Ф.** Аналитический подход к формированию конфигураций технических систем // Автоматика и телемеханика. 2017. № 9. С. 67—83.
16. **Гамаюнов И. Ф.** Генерирование альтернативных решений в задаче управления избыточностью технических комплексов // Автоматика и телемеханика. 2018. № 4. С. 92—104.
17. **Агеев А. М., Бронников А. М., Буков В. Н., Гамаюнов И. Ф.** Супервизорный метод управления технических систем с избыточностью // Изв. РАН. Теория и системы управления. 2017. № 3. С. 72—82.
18. **Буков В. Н., Шурман В. С., Гамаюнов И. Ф., Агеев А. М.** Управление избыточностью вычислительных ресурсов интегрированной модульной авионики // Мехатроника, автоматизация, управление. 2019. Т. 20, № 6. С. 376—384.
19. **Дьяконов В. П.** MATLAB 7.*/R2006/R2007: Самоучитель. М.: ДМК Пресс, 2008. 768 с.
20. **Hunt B. R., Lipsman R. L., Rosenberg J. M.** MatLab R2007 с нуля! Книга + Видеокурс. М.: Лучшие книги, 2008. 352 с.

A Software Tool to Support the Synthesis of a Redundancy Management System for an Aircraft Equipment Complex

V. N. Bukov, v_bukov@mail.ru, Open Joint-Stock Company "Airborne Navigation Systems", Moscow, 127015, Russian Federation, **A. M. Ageev**, ageev_bbc@mail.ru, **A. M. Maltsev**, max_alex_67@mail.ru, Military Training and Scientific Center of the Air Force "Air Force Academy named after prof. N. E. Zhukovsky and Yu. A. Gagarin", Voronezh, 394064, Russian Federation

Corresponding author:

Bukov Valentin N., D. Sc., Chief Scientific Researcher, Open Joint-Stock Company "Airborne Navigation Systems", Moscow, 127015, Russian Federation
E-mail: v_bukov@mail.ru

Received on November 13, 2019
Accepted on December 13, 2019

Computer tools for the development of onboard equipment complexes of technical systems with managed redundancy are considered. These tools cover such phases as setting the initial data, including the nominal configuration, generating alternative configurations, and creating a table of configurations of the redundant complex. A methodological example is given.

Keywords: *integrated onboard equipment complex, integrated modular avionics, redundancy management, alternative configuration generation, configuration supervisors*

For citation:

Bukov V. N., **Ageev A. M.**, **Maltsev A. M.** A Software Tool to Support the Synthesis of a Redundancy Management System for an Aircraft Equipment Complex, *Programmnaya Ingeneria*, 2020, vol. 11, no. 2, pp. 96—107

DOI: 10.17587/prin.11.106-117

References

1. **DO-297.** Integrated modular avionics (IMA) development guidance and certification considerations. Washington, RTCA Inc., 2005.

2. **Watkins C. B.** Integrated Modular Avionics: Managing the Allocation of Shared Intersystem Resources, *25th Digital Avionics Systems Conference (DASC),* 15—19 October 2006, Portland, Oregon, USA, available at: https://ieeexplore.ieee.org/document/4106349

3. **Garside R., Pighetti J. F.** Integrating Modular Avionics: A New Role Emerges, *26th Digital Avionics Systems Conference (DASC),* 21—25 October 2007, Dallas, Texas, USA, available at: https://ieeexplore.ieee.org/document/4391843

4. **Hainaut D.** Towards the Next Generation of Integrated Modular Avionics, *Aerodays,* 30 March - 01 April 2011, Madrid, Spain, available at: http://www.pplane-project.org/Download/News/Info/document/1007.pdf

5. **Bernard S., Garcia J.-P.** Braking Systems with New IMA Generation, *Aerospace Technology Conference and Exposition. SAE International*, October 2011, paper 2011-01-2662. DOI: 10.4271/2011-01-2662.

6. **Buzdalov D. V., Zelenov S. V., Kornikhin E. V., Petrenko A. K., Fear A. V., Ugnenko A. A., Khoroshilov A. V.** Tools for Designing Integrated Modular Avionics Systems, *Trudy Instituta sistemnogo programmirovaniya RAN*, 2014, vol. 26, issue 1, pp. 201—230 (in Russian).

7. **Degtyarev A. R., Medvedev G. V.** The Distribution Algorithm of Tasks in Multi-Processor Complexes of Integrated Modular Avionics, *Avtomatizaciya processov upravleniya*, 2014, no. 1 (35), pp. 79—84 (in Russian).

8. **Bukov V., Kutahov V., Bekkiev A.** Avionics of Zero Maintenance Equipment, *27th Cong. of the Int. Council of the Aeronaut. Sciences*, 19—24 September 2010, Nice, France, 2010, paper 7-1-1.

9. **Averyanov A. V., Baranovskii A. M., Esaulov K. A.** Determination of limits of hardware redundancy of control systems, *Izvestiya vysshih uchebnyh zavedenij. Priborostroenie,* 2014, vol. 57, no. 3, pp. 23—26 (in Russian).

10. **Shulga T. E.** General control scheme for discrete functions based on functional redundancy, *Sovremennye tekhnologii. Sistemnyj analiz. Modelirovanie,* 2010, no. 1. pp. 167—174 (in Russian).

11. **Vasenin V. A., Pirogov M. V., Chechkin A. V.** The main operator of radical modeling on a demo, *Programmnaya ingeneria*, 2016, vol. 7, no. 2, pp. 75—85 (in Russian).

12. **Rudnev Yu. P., Khetagurov Ya. A.** *Improving the reliability of digital devices using redundant coding methods,* Moscow, Energia, 1974, 272 p. (in Russian).

13. **Bukov V. N., Bronnikov A. M., Ageev A. M., Gamayunov I. F., Ozerov E. V., Shurman V. A.** The concept of controlled redundancy of complexes of onboard equipment, *Nauch. chteniya po aviacii, posv. pam. N. E. Zhukovskogo: Mater. XVI Vseros. nauch.-prakt. konf.*, eds S. P. Khalyutin. Moscow, Publishing House Acad. them. N. E. Zhukovsky, 2019, pp. 17—33 (in Russian).

14. **Ageev A. M.** Configuration of Redundancy Onboard Equipment Complexes, *J. Comput. Syst. Sci. Int.,* 2018, vol. 57, issue 4, pp. 640—654.

15. **Bukov V. N., Bronnikov A. M., Ageev A. M., Gamayunov I. F.** An analytic approach to constructing configurations of technical systems, *Automation and Remote Control,* 2017, vol. 78, issue 9, pp. 1600—1613.

16. **Gamayunov I. F.** Generation of Alternative Solutions in the Redundancy Management Problem for Hardware Complexes, *Automation and Remote Control,* 2018, vol. 79, issue 4, pp. 655—664.

17. **Ageev A. M., Bronnikov A. M., Bukov V. N., Gamayunov I. F.** Supervisory control method for redundant technical systems, *J. Comp. and Syst. Sciences Int.,* 2017, vol. 56, no. 3, pp. 410—419.

18. **Bukov V. N., Shurman V. S., Gamayunov I. F., Ageev A. M.** Management of the redundancy of computing resources of integrated modular avionics, *Mekhatronika, avtomatizatsya, upravlenie*, 2019, vol. 20, no. 6, pp. 376—384 (in Russian).

19. **Dyakonov V. P.** *MATLAB 7. */R2006/R2007: Tutorial*, Moscow, DMK Press, 2008, 768 p. (in Russian).

20. **Hunt B. R., Lipsman R. L., Rosenberg J. M.** *MatLab R2007 from scratch! Book + Video course,* Moscow, Luchshie knigi, 2008, 352 p. (in Russian).

ИНФОРМАЦИЯ

1—5 июня 2020 г., Казанский национальный исследовательский технологический университет (КНИТУ), Казань, Россия

5—10 октября 2020 г., Калининградский государственный технический университет (КГТУ), Калининград, Россия

14—16 октября 2020 г., Саратовский государственный технический университет имени Гагарина Ю. А., Саратов, Россия

26—30 октября 2020 г., Белорусский национальный технический университет (БНТУ), Минск, Беларусь

Распределенная во времени и пространстве

XXXIII Международная научная конференция

"МАТЕМАТИЧЕСКИЕ МЕТОДЫ В ТЕХНИКЕ И ТЕХНОЛОГИЯХ — ММТТ-33"

Подробная информация о конференции и условиях участия в ней размещается на сайте http://mmtt.sstu.ru/

УДК 004.5 DOI: 10.17587/prin.11.108-114

Д. И. Читалов, мл. науч. сотр., cdi9@yandex.ru, Федеральное государственное бюджетное учреждение науки Южно-Уральский федеральный научный центр минералогии и геоэкологии Уральского отделения Российской академии наук, Россия, Челябинская обл., г. Миасс, Ильменский заповедник

О разработке модуля для реализации движения и топологического изменения расчетных сеток и его интеграции в графическую оболочку для платформы OpenFOAM

Освещены аспекты проектирования программного модуля, обеспечивающего возможность корректировки геометрии расчетных сеток путем реализации их движения и топологического изменения в рамках проведения численных экспериментов в области механики сплошных сред с использованием платформы OpenFOAM. Даны диаграммы структуры и логики этого модуля, а также стек технологий, на основе которого он реализован. Представлены результаты испытаний модуля на примере одной из стандартных задач механики сплошной среды, входящих в дистрибутив OpenFOAM. Сформулированы итоговые выводы, новизна и практическая ценность исследования.

Ключевые слова: *численное моделирование, механика сплошных сред, графический интерфейс пользователя, OpenFOAM, язык программирования Python 3.5, открытое программное обеспечение, утилита moveDynamicMesh, библиотека PyQt5, СУБД SQLite*

Введение

Данная статья продолжает исследование, направленное на разработку отечественной графической оболочки (GUI) для упрощения процедур численного моделирования на базе платформы OpenFOAM [1]. Эта платформа имеет статус одного из наиболее известных свободно распространяемых программных комплексов, используемых на предприятиях сферы машиностроения для численного моделирования объектов и процессов. Автором разработана и размещена в открытом доступе для интернет-пользователей базовая версия графической оболочки, созданная в 2016 г. и дополненная в последующие годы новыми модулями. Такие модули расширили возможности пользователя по выполнению этапа препроцессинга (подготовки проекта задачи механики сплошной среды — МСС к решению [2—5]) в его части, именуемой формированием расчетной сетки (РС). Под проектом задачи МСС (расчетным случаем) подразумевается директория, содержащая набор служебных файлов-словарей, определяющих расчетные параметры.

Платформа OpenFOAM совершенствуется, и к ее версии 2.1 перед пользователем данного программного продукта открылись новые возможности. Они касаются не только непосредственно численного решения задачи МСС, но и выполнения ряда других важных задач, определяющих точность проводимого эксперимента. Результаты данного исследования призваны дополнить исходный код графической оболочки еще одним модулем. Его функция — автоматизировать один из шагов подготовки РС, а именно реализацию движения и технологического изменения РС. Такая задача на платформе OpenFOAM решается с помощью утилиты moveDynamicMesh. Разработанная графическая оболочка, в том числе и представленный программный модуль в настоящее время тестируются на одном из крупнейших российских ракетостроительных предприятий — АО ГРЦ им. Макеева [6]. В перспективе они могут быть внедрены и на предприятиях других отраслей машиностроения.

Усовершенствованная версия графической оболочки (с интеграцией описываемого модуля) может эффективно использоваться в рамках численного моделирования на базе OpenFOAM наряду с графическими оболочками Salome [7], Helyx-OS [8], Visual-CFD [9] и даже превосходить их по некоторым показателям. Следует отметить, что платформа OpenFOAM считается одним из лучших аналогов популярного коммерческого решения ANSYS [10], которое для небольших предприятий является дорогостоящим приобретением.

Настоящее исследование обобщает результаты очередного этапа доработки графической оболочки [2] в части обеспечения доступа специалиста к утилитам этапа препроцессинга, обеспечивающим модификацию РС. Результаты этого исследования

позволяют сделать РС более точной и, как следствие, повысить точность конечной численной модели. Поскольку графическая оболочка, представленная в работе [2], реализована по модульному принципу, предложено разработать дополнительный модуль, расширяющий ее исходный код и позволяющий обеспечивать движение и топологическое изменение РС.

Модуль должен предусматривать наличие графических средств подготовки соответствующих служебных файлов-словарей (в данном случае это файл-словарь dynamicMeshDict) и средств запуска утилиты moveDynamicMesh, которая отвечает за реализацию движения и топологического изменения РС. Необходимость разработки данного модуля обусловлена тем обстоятельством, что специалистам АО ГРЦ им. Макеева оказалось недостаточно стандартных возможностей графической оболочки для решения задач в области проектирования. В частности, этот модуль оказался востребованным при решении задачи построения РС с учетом большего числа характеристик реального объекта и процесса.

Благодаря использованию графической оболочки [2] совместно с описываемым модулем сокращаются затраты рабочего времени специалистов на модификацию сеточных моделей. Отметим, что освоение модуля пользователями не требует существенных усилий. Его программное решение сопровождается русскоязычной документацией, а графическая часть (интерфейс) реализована с возможностью переключения с англоязычной версии на русифицированную. В рамках исследования по разработке модуля автором изучена официальная документация пользователя платформы OpenFOAM [11], а также пособие с практическими примерами [12].

Назначение утилиты moveDynamicMesh

Утилита moveDynamicMesh предназначена для манипулирования РС и используется во всех численных экспериментах, которые связаны с необходимостью движения РС. Утилита предусматривает две категории операций для модификации РС: автоматическое движение и топологическое изменение РС.

Обе категории операций осуществляются на этапе препроцессинга после генерации сеточных моделей на базе встроенных в платформу OpenFOAM утилит, а именно — blockMesh, snappyHexMesh и др. [3—5]. Для указания параметров движения и топологического изменения от пользователя требуется создать и заполнить служебный файл-словарь dynamicMeshDict, который должен находиться в поддиректории constant директории проекта задачи МСС. В таблице приведены основные параметры для реализации манипуляций с движением сеточной модели.

Под изменением топологии подразумевается одна из основных операций с геометрией и сеткой, а именно — изменение нумерации узлов. Она позволяет проводить численный эксперимент и моделировать большее число реальных ситуаций, связанных с исследуемой задачей МСС, т. е. учитывать больше исходных данных.

Таблица содержит описания параметров и примеры их значений для учебной задачи МСС propeller, входящей в стандартный дистрибутив платформы OpenFOAM. Решение этой задачи моделируется с помощью решателя interPhaseChangeDyMFoam, ориентированного на эксперименты в области многофазного потока. Информация о других возможных значениях перечисленных параметров доступна по ссылке [13].

Основной параметр словаря dynamicMeshDict — это dynamicFvMesh. Значением параметра может быть один из классов, обеспечивающих движение РС, например, dynamicMotionSolverFvMesh. Этот класс-решатель позволяет изменить сеточную модель вокруг заданного набора границ. Движение РС определяется через давление на этих границах. При этом решатель dynamicMotionSolverFvMesh реализует обратную связь для моделирования процессов в жид-

Параметры файла-словаря dynamicMeshDict

Параметр	Описание	Пример значения
dynamicFvMesh	Определяет программу-решатель для обеспечения автоматического движения РС	dynamicMotionSolverFvMesh
topoChangerFvMesh	Определяет программу-решатель для топологического изменения РС	dynamicMotionSolverTopoFvMesh
motionSolverLibs	Определяет РС в качестве динамической	libfvMotionSolvers.so
motionSolver	Определяет основную программу-решатель для реализации движения РС	solidBody
cellZone	Определяет область ячейки	innerCylinderSmall
solidBodyMotionFunction	Определяет движение твердого тела	rotatingMotion
origin	Определяет исходную точку	(0 0 0)
axis	Определяет ось движения	(0 1 0)
omega	Определяет омега-модель	table

костях. Он модифицирует граничные условия скорости на включенных границах для указания локальной скорости моделируемого тела. Эта локальная скорость включает все допустимые манипуляции, в частности, поступательное и вращательное движения. Такой контроль сетки, как правило, применяется в контексте вопросов, связанных с движением твердого тела.

На практике в качестве значения параметра dynamicFvMesh можно также встретить класс-решатель dynamicTopoFvMesh, изменяющий топологию сетки при наличии значительных деформаций, а также позволяющий добавлять и удалять слои РС.

Постановка цели и задач исследования

Цель настоящего исследования заключается в изучении одного из механизмов модификации сеточных моделей, реализованного в платформе OpenFOAM, а также в разработке программного модуля, автоматизирующего этот механизм. Его использование позволяет численно решать задачи МСС с большей точностью и учитывать большее число исходных параметров таких задач. Суть механизма заключается в реализации движения и топологического изменения РС посредством утилиты moveDynamicMesh.

Для освоения механизмов утилиты moveDynamicMesh, работа с которой традиционно (без использования GUI) осуществляется посредством командной строки, автором проанализированы специфика и требования к ее использованию. Они касаются подготовки служебного файла-словаря dynamicMeshDict, содержащего основные характеристики, определяющие особенности модификации РС с помощью утилиты moveDynamicMesh. К этим характеристикам относятся: параметры, обеспечивающие контроль формирования сетки; физические параметры моделируемого твердого тела; параметры, определяющие движение тела, а также величины сил и ограничения движения, в дополнение к силам, действующим со стороны жидкости.

Предложенный программный модуль расширяет исходный код графической оболочки rCF_GUI, созданной автором и используемой в качестве интерфейса для численных экспериментов на базе платформы OpenFOAM [2]. Расширение исходного кода предполагает создание необходимых графических элементов (экранных форм), а также доработку механизма программного запуска консольных утилит платформы OpenFOAM, в частности, утилиты moveDynamicMesh. Предложенные изменения направлены на упрощение работы с утилитой и файлом-словарем dynamicMeshDict. Таким образом, доработка автором текущей версии графической оболочки rCF_GUI потребовала решения перечисленных далее задач.

1. Реализовать макет экранной формы, "связывающей" пользователя с файлом-словарем dynamicMeshDict. Разработать и реализовать механизм записи параметров из формы в указанный файл.

2. Дополнить инструментальную панель графической оболочки кнопкой открытия добавленной экранной формы.

3. Обеспечить вывод параметров файла-словаря dynamicMeshDict в окно отображения результатов графической оболочки.

4. Реализовать механизм сериализации определяемых в форме параметров и их последующего восстановления для обеспечения возможности редактирования файла-словаря dynamicMeshDict.

5. Реализовать для элементов управления экранной формы набор валидаторов, контролирующих корректность типов и формата указываемых данных.

Средства разработки

Программный код описываемого модуля расширяет исходный код базовой версии графической оболочки rCF_GUI [2]. Для реализации исходного кода модуля используются инструментальные средства, к числу которых относятся перечисленные далее.

1. Язык программирования Python 3.5 [14]. Согласно обновленному индексу TIOBE входит в тройку самых востребованных средств описания логики программных приложений [15].

2. Фреймворк PyQt5. Популярная библиотека, позволяющая создавать привычные оконные интерфейсы для Python-приложений [16, 17]. Имеет подробную документацию с примерами и многочисленное сообщество пользователей.

3. Система хранения данных SQLite. Интегрирована в стандартную библиотеку Python. Поддерживает как протокол DB-API, позволяющий работать с базой данных через традиционные SQL-запросы, так и возможность подключения ORM-библиотек, например, SQLAlchemy.

4. Среда разработки PyCharm Community Edition. Открытая версия одной из самых известных интегрированных сред для написания, отладки и запуска кода. Имеет встроенные механизмы подсветки кода, подсказок, приведения кода к стандарту PEP-8, создания виртуального окружения для проекта.

Все перечисленные инструментальные средства являются свободно распространяемыми и позволяют на их основе разрабатывать открытые программные решения, не требующие приобретения лицензии.

Усовершенствованная версия графической оболочки (с интегрированным модулем для модификации РС), как и первоначальная версия приложения [2] функционирует на вычислительных устройствах под управлением ОС Linux. При этом должна быть инсталлирована и платформа OpenFOAM, отвечающая за весь ход процесса численного моделирования, а также среда визуализации результатов научных экспериментов ParaView [18].

Структура и логика работы модуля

Базовая версия графической оболочки rCF_GUI [2] реализована по модульному принципу, что позволяет при необходимости расширять ее возможности путем интеграции в нее дополнительных модулей. На рис. 1, 2 представлены диаграммы, описывающие структуру и логику работы усовершенствованной версии графической оболочки (с интегрированным

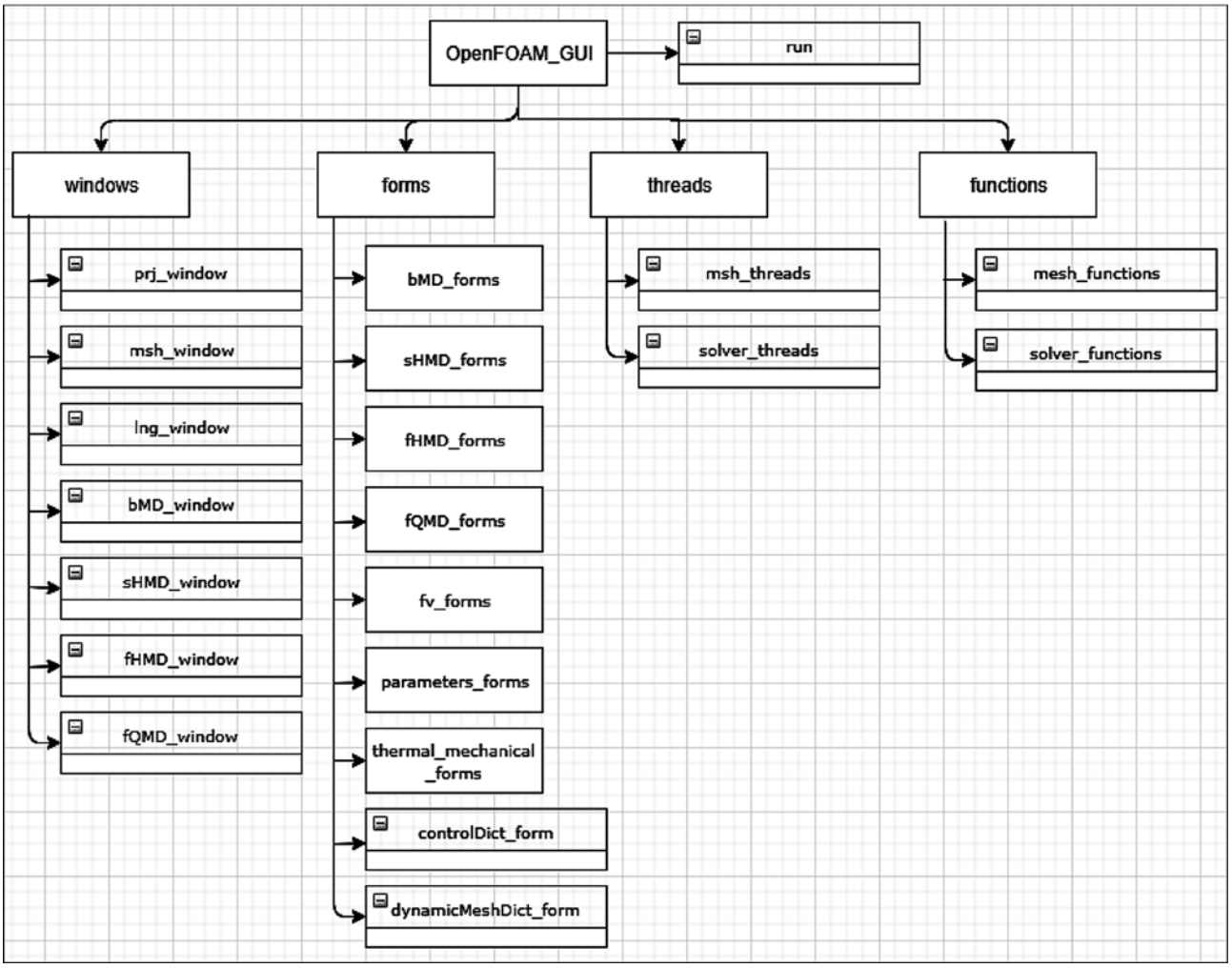

Рис. 1. Структура графической оболочки OpenFOAM_GUI

модулем). Усовершенствованной версии приложения присвоено имя OpenFOAM_GUI.

Файлы с исходным кодом графической оболочки распределены по четырем директориям, расположенным в папке — корне приложения. В директории windows находятся файлы-модули, программный код которых определяет структуру основных окон интерфейса, например, окна выбора расчетного случая, окна для определения параметров РС. Директория forms включает набор вложенных папок с файлами, каждый из которых определяет элементы управления для указания параметров отдельных блоков файлов-словарей с параметрами задачи МСС. В папке threads находятся программные модули, обеспечивающие реализацию многопоточности в процессе численного решения задачи МСС. Директория functions включает файлы-модули с программным кодом служебных функций приложения. За запуск графической оболочки отвечает файл run.py, расположенный в папке — корне проекта.

Работа с рассматриваемым в статье модулем осуществляется на этапе препроцессинга численного эксперимента, когда определяются параметры геометрии расчетной области, а также генерируется РС с помощью консольных утилит blockMesh, snappyHexMesh, foamyHexMesh, foamyQuadMesh [3—5]. Результат визуализации сгенерированной РС может соответствовать требованиям, предъявляемым к моделируемой задаче. В этом случае пользователь переходит к следующим этапам численного эксперимента. При несоответствии требованиям специалист может выполнить модификацию РС, использовав возможности модуля по реализации движения и топологического изменения РС. Все перечисленные выше шаги пользователь осуществляет через главное окно графической оболочки. На любом шаге работы с РС допустимо возвращение к редактированию ее параметров и повторный запуск процесса визуализации.

На рис. 3 (см. вторую сторону обложки) представлено главное окно приложения OpenFOAM_GUI после определения параметров движения и топологического изменения РС с визуализацией результатов. В качестве примера использована уже упомянутая ранее задача МСС propeller, моделируемая посредством OpenFOAM-решателя interPhaseChangeDyMFoam (предназначен для проведения экспериментов в области многофазного потока).

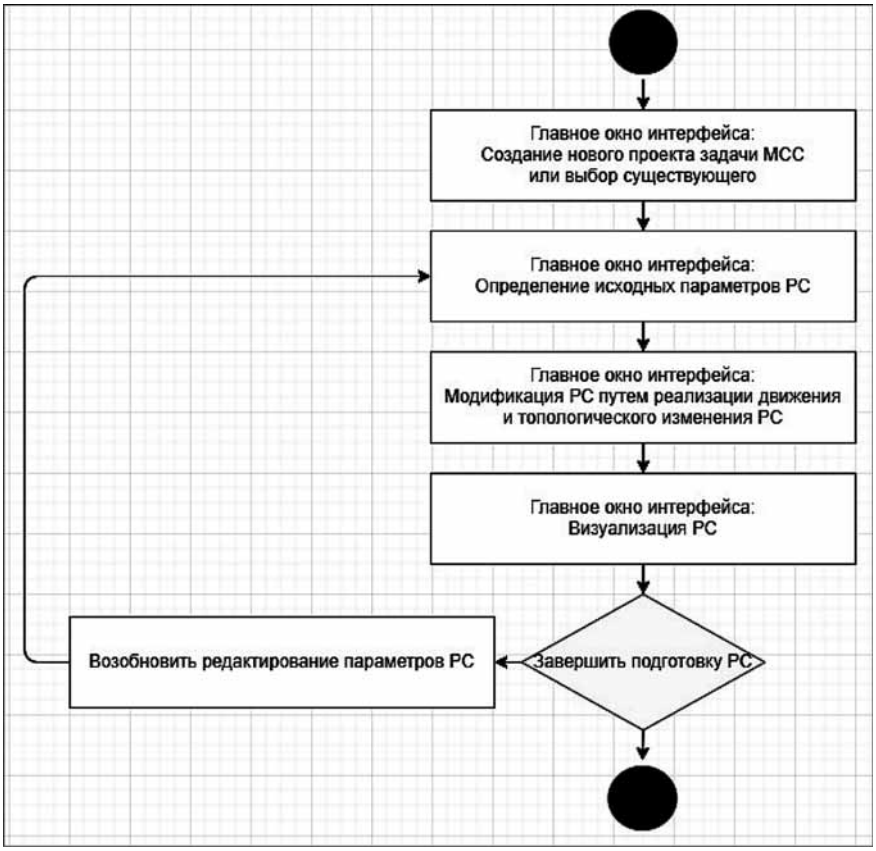

Рис. 2. Логика работы графической оболочки OpenFOAM_GUI в части модификации РС с помощью утилиты moveDynamicMesh

Результаты исследования

Итогом осуществленного автором исследования является расширение исходного кода графической оболочки OpenFOAM_GUI для обеспечения возможности автоматизации выполнения одного из шагов препроцессинга численного решения. В качестве такого шага рассматривается реализация движения и топологического изменения РС для повышения точности численного расчета решения задачи МСС и построения численной модели на большем спектре исходных данных (утилита moveDynamicMesh платформы OpenFOAM).

В результате проведенной работы модифицировано главное окно графической оболочки: усовершенствовано окно вывода файловой структуры проекта задачи МСС (см. рис. 3 на второй стороне обложки). Теперь у пользователя существует возможность работы и с графическими средствами для заполнения файла-словаря dynamicMeshDict. Разработаны и интегрированы программные механизмы генерации данного файла на основе введенных пользователем данных. Также реализован алгоритм запуска утилиты moveDynamicMesh из главного окна графической оболочки.

В задаче МСС propeller, на примере которой рассматривается применение графической оболочки, реализуется численная модель морского винта. На основе этой модели определяются характеристики объекта под воздействием окружающих сред. Результаты такого эксперимента находят применение в конструкторских подразделениях машиностроительных предприятий при проектировании продукции. На рис. 3 (см. вторую сторону обложки) представлен пример расчетной области, которая может быть сформирована в рамках данного эксперимента и модифицирована с помощью утилиты moveDynamicMesh.

Заключение

Исследование, описанное в настоящей статье, посвящено изучению особенностей использования утилиты dynamicMeshFoam для реализации программных механизмов, обеспечивающих движение и топологическое изменение РС для численных моделей МСС. Такие манипуляции реализуются с помощью инструментальных средств платформы OpenFOAM и разработки модуля, позволяющего автоматизировать эту задачу и расширяющего исходный код разработанной автором графической оболочки rCF_GUI [2]. Представленный программный модуль упрощает работу специалиста с утилитой dynamicMeshFoam и упрощает подготовку служебного файла-словаря dynamicMeshDict, определяющего параметры движения и изменения РС.

Автором разработаны следующие средства, реализованные в представленном модуле:

1) экранная форма для файла-словаря dynamicMeshDict с элементами управления, позволяющими пользователю указать параметры движения и изменения РС;

2) механизм генерации и запуска bash-скрипта, обеспечивающего работу утилиты dynamicMeshFoam;

3) система валидаторов, контролирующих корректность типов и форматов значений, задаваемых через экранную форму файла-словаря dynamicMeshDict;

4) механизм сериализации и восстановления параметров файла-словаря dynamicMeshDict для обеспечения корректировки параметров движения и топологического изменения РС для существующих проектов задач МСС;

5) механизм подготовки нескольких версий служебного файла-словаря dynamicMeshDict для одной задачи МСС, позволяющий выполнять численный эксперимент с учетом различных исходных условий.

К достоинствам представленного модуля можно отнести предполагаемое снижение затрат рабочего времени при работе с утилитой dynamicMeshFoam и минимизацию ошибок в ходе препроцессинга. Благодаря графическому интерфейсу модуля и встроенному программному алгоритму специалист тратит меньше времени на подготовку служебного файла dynamicMeshDict и запуск утилиты dynamicMeshFoam.

Трудоемкий и затратный по времени подход к работе через командную строку заменяется на более быстрый и эффективный — использование привычного оконного интерфейса с экранными формами и элементами управления. Кроме того, применение валидаторов позволяет избежать возможных ошибок при указании типов данных параметров, т. е. избежать ошибок при построении сеточной модели.

Описанный программный модуль, как и сама графическая оболочка разработаны на базе одного из самых востребованных языков программирования Python 3.5 и фреймворка для проектирования оконных интерфейсов PyQt5. Набор кода, его отладка и тестирование выполнены с помощью популярной интегрированной среды разработки PyCharm Community Edition. Созданная версия графической оболочки (OpenFOAM_GUI) распространяется как открытое ПО и развернута на сервисе хостинга IT-проектов GitHub [20].

Таким образом, разработанный модуль, интегрированный в текущую версию графической оболочки rCF_GUI, расширяет исходный код платформы OpenFOAM. Усовершенствованная версия графической оболочки может применяться исследователями и инженерами при моделировании задач МСС на предприятиях большинства отраслей промышленности.

Список литературы

1. **OpenFOAM.** The open source CFD toolbox. URL: https://www.openfoam.com/
2. **Читалов Д. И., Меркулов Е. С., Калашников С. Т.** Разработка графического интерфейса пользователя для программного комплекса OpenFOAM // Программная инженерия. 2016. Т. 7, № 12. С. 568—574. DOI: 10.17587/prin.7.568-574.
3. **Читалов Д. И., Калашников С. Т.** Разработка приложения для подготовки расчетных сеток с градуирующими и изогнутыми краями для программной среды OpenFOAM // Системы и средства информатики. 2018. Т. 28, № 4. С. 122—135. DOI: 10.14357/08696527180412
4. **Читалов Д. И., Калашников С. Т.** Разработка приложения для подготовки расчетных сеток посредством утилиты snappyHexMesh программной среды OpenFOAM // Программные продукты и системы. 2018. Т. 31, № 4. С. 715—722. DOI: 10.15827/0236-235X.124.715-722
5. **Читалов Д. И., Калашников С. Т.** Разработка приложения для подготовки расчетных сеток с помощью утилиты foamyQuadMesh платформы OpenFOAM // Программная инженерия. 2018. Т. 9, № 7. С. 311—317. DOI: 10.17587/prin.9.311-317.
6. **Государственный** ракетный центр имени академика В. П. Макеева. URL: http://www.makeyev.ru/
7. **SALOME**. URL: http://www.salome-platform.org
8. **HELYX-OS.** URL: http://engys.com/products/helyx-os
9. **Visual-CFD** for OpenFOAM. URL: https://www.esi-group.com/software-solutions/virtual-environment/cfd-multiphysics/visual-cfd-openfoam
10. **ANSYS.** URL: https://www.ansys.com/
11. **OpenFOAM**. User Guide. URL: http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf
12. **OpenFOAM.** Tutorial Guide. URL: https://www.openfoam.com/documentation/tutorial-guide/index.php
13. **Parameter** Definitions — dynamicMotionSolverFvMesh. URL: https://openfoamwiki.net/index.php/Parameter_Definitions_-_dynamicMotionSolverFvMesh
14. **Python 3.5** documentation. URL: https://docs.python.org/3.5/
15. **TIOBE** Index https://www.tiobe.com/tiobe-index/
16. **PyQt5** Reference Guide. URL: http://pyqt.sourceforge.net/Docs/PyQt5/
17. **Прохоренок Н. А.** Python 3 и PyQt. Разработка приложений. СПб.: БХВ-Петербург, 2012. 704 с.
18. **ParaView.** URL: https://www.paraview.org/
19. **Advanced** bash-scripting guide. URL: https://www.tldp.org/LDP/abs/html/
20. **OpenFOAM_GUI**. URL: https://github.com/DmitryChitalov/OpenFOAM_GUI

On the Development of a Module for Implementing Motion and Topological Changes in Computational Meshes and its Integration into the Graphical Shell for the OpenFOAM Platform

D. I. Chitalov, cdi9@yandex.ru, South Urals Federal Research Centre of Mineralogy and Geoecology of the UB RAS, Miass, Ilmen reserve, 456317, Chelyabinsk Region, Russian Federation

Corresponding author:

Chitalov Dmitry I., Junior Researcher, South Urals Federal Research Centre of Mineralogy and Geoecology of the UB RAS, Miass, Ilmen reserve, 456317, Chelyabinsk Region, Russian Federation
E-mail: cdi9@yandex.ru

Received on January 21, 2020
Accepted on February 10, 2020

This article highlights aspects of designing a software module that provides the ability to adjust the geometry of computational meshes by realizing their motion and topological changes as part of numerical experiments in the field of continuum mechanics using the OpenFOAM platform. This type of modification of the mesh model is carried out as part of the preprocessing of a numerical experiment using the moveDynamicMesh utility and allows you to take into account a larger number of parameters of a real object or process in the numerical model. The module is integrated into the basic version of the graphical shell for working with the OpenFOAM platform, thereby expanding the

capabilities of the graphical shell and making it effective for solving a larger list of problems in continuum mechanics. The relevance of the study is determined, the shortcomings of existing solutions are identified. A description of the moveDynamicMesh utility, the features of the formation of the corresponding dictionary file, and a list of its possible parameters that determine the characteristics of the numerical model are given. At the beginning of the article, the goal of the study and the set of tasks necessary for its achievement are formulated. The diagrams of the structure and logic of the module, as well as descriptions of the diagrams are given. A stack of technologies is presented, on the basis of which a module is implemented, including the language for implementing the logic of its operation and a framework for designing a graphic component. The results of module tests are presented on the example of one of the standard tasks of continuum mechanics included in the OpenFOAM distribution. The final conclusions, novelty and practical value of the study are formulated. The source code of the module is hosted on the hosting service of GitHub IT projects, the corresponding link is provided.

Keywords: *numerical simulation, continuum mechanics, graphical user interface, OpenFOAM, Python 3.5, open source software, moveDynamicMesh utility, PyQt5, SQLite*

For citation:

Chitalov D. I. On the Development of a Module for Implementing Motion and Topological Changes in Computational Meshes and its Integration into the Graphical Shell for the OpenFOAM Platform, *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 108—114

DOI: 10.17587/prin.11.108-114

References

1. **OpenFOAM.** The open source CFD toolbox, available at: http://www.openfoam.com/
2. **Chitalov D. I., Merkulov E. S., Kalashnikov S. T.** Development of a graphical user interface for the OpenFOAM toolbox, *Programmnaya ingeneria*, 2016, vol. 7, no. 12, pp. 568—574 (in Russian).
3. **Chitalov D. I., Kalashnikov S. T.** Development of an application for the preparation of computational meshes with graduating and curved edges for the OpenFOAM software, *Systemi i sredstva informatiki*, 2018, vol. 28, no. 4, pp. 122-135 (in Russian).
4. **Chitalov D. I., Kalashnikov S. T.** Development of an application for preparing calculation grids using the snappyHexMesh utility of the OpenFOAM software environment, *Programmnie producti i sistemi*, 2018, vol. 31, no 4, pp. 715—722 (in Russian).
5. **Chitalov D. I., Kalashnikov S. T.** Application development for meshes preparation using foamyQuadMesh utility for the OpenFOAM toolbox, *Programmnaya ingeneria*, 2018, vol. 9, no 7, pp. 311—317 (in Russian).
6. **Makeyev** SRC, available at: http://www.makeyev.ru/
7. **Salome.** The Open Source Integration Platform for Numerical Simulation, available at: http://www.salome-platform.org
8. **Helyx-OS**. Open Source GUI for OpenFOAM, available at: http://engys.com/products/helyx-os
9. **Visual-CFD** for OpenFOAM. CFD simulation software aimed at solving complex flow applications, available at: http://www.esi-group.com/software-solutions/virtual-environment/cfd-multiphysics/visual-cfd-openfoam
10. **ANSYS**, available at: http://www.ansys.com/
11. **The OpenFOAM** Foundation. User Guide, available at: http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf
12. **OpenFOAM.** Tutorial Guide, available at: http://www.openfoam.com/documentation/tutorial-guide/index.php
13. **dynamicMotionSolverFvMesh,** available at: http://openfoam-wiki.net/index.php/Parameter_Definitions_-_dynamicMotionSolverFvMesh.
14. **Python 3.5** documentation, available at: http://docs.python.org/3.5/
15. **Tiobe** Index, available at: http://www.tiobe.com/tiobe-index/
16. **PyQt5** Reference Guide, available at: http://pyqt.sourceforge.net/Docs/PyQt5/
17. **Prohorenok N. A.** *Python 3 and PyQt. Application Development*, Saint-Petersburg, BHV-Petersburg, 2012, 704 p. (in Russian).
18. **ParaView.** Open-source, multi-platform data analysis and visualization application, available at: http://www.paraview.org/
19. **Advanced** bash-scripting guide, available at: http://www.tldp.org/LDP/abs/html/
20. **OpenFOAM_GUI,** available at: http://github.com/Dmitry-Chitalov/OpenFOAM_GUI

ИНФОРМАЦИЯ

Международный конгресс по кибербезопасности 2020

Даты проведения: 09.07.2020—10.07.2020 г.

Место проведения: Россия, Москва

Конгресс — одно из ключевых событий года в сфере кибербезопасности, международная межотраслевая площадка для диалога представителей органов государственной власти, лидеров мирового бизнеса и ведущих экспертов по самым острым и актуальным вопросам защиты от киберугроз в эпоху глобальной цифровой трансформации

Подробности: https://icc.moscow/ru/

УДК 004.658 DOI: 10.17587/prin.11.115-122

Р. Э. Асратян, канд. техн. наук, ст. науч. сотр., вед. науч. сотр., rea@ipu.ru,
Институт проблем управления им. В. А. Трапезникова РАН, Москва

Служба плановой обработки данных в СУБД PostgreSQL для среды Linux

Рассмотрены основы организации и функционирования служебной фоновой программы (демона) PdbCron, предназначенной для выполнения периодической плановой обработки данных в СУБД PostgreSQL (чистку временных или устаревших данных, проверку корректности или резервного копирования данных) по собственному расписанию в автоматическом режиме в операционной среде Linux.

Ключевые слова: *информационные системы, базы данных, СУБД PostgreSQL, ОС Linux, расписание обработки*

Введение

В последние годы разработчики информационных систем все в большей степени обращают свое внимание на ОС Linux [1, 2] и СУБД PostgreSQL [3, 4], как на среду и средство поддержки их функционирования. Это связано не только с естественным интересом к свободно распространяемому программному обеспечению, но также с постоянным улучшением качества и характеристик этих продуктов. Тем не менее, пользователи некоммерческих версий PostgreSQL в настоящее время сталкиваются с определенными трудностями, связанными с организацией плановых работ по контролю и поддержке баз данных (БД). К числу таких относятся: чистка временных или устаревших данных; проверка корректности; генерация дампов (резервных копий) данных и т. п. Отсутствие доступных средств обработки БД по заданному расписанию заставляет разработчиков искать собственные технические решения, основанные на использовании общесистемного планировщика заданий Cron и, как правило, приспособленные к нуждам конкретного проекта. Ситуация осложняется еще и тем, что структура расписания в Cron не свободна от недостатков (воспроизведенных и в коммерческом продукте Postgres Scheduler [5]). Например, в нем отсутствует понятие недели (номера недели в году), хотя эта единица времени играет важную роль в деловых приложениях. Поэтому временная привязка вида "выполнять каждую вторую неделю по четвергам в полночь" выражается на "языке" Cron чрезвычайно громоздко. То же самое можно сказать о простейшем правиле: "выполнять через каждые 20 дней", невзирая на границы месяцев.

Описываемая в данной работе программа PdbCron представляет собой попытку поиска "общего решения" упомянутых выше вопросов. Она предназначена для выполнения периодической обработки данных в СУБД PostgreSQL по собственному расписанию без участия человека. Как и программа Cron, она представляет собой постоянно активную программу (демон) в среде ОС Linux. Эта программа целиком сфокусирована на обработке БД, а не на планировании заданий в ОС.

Расписание PdbCron

Уметь пользоваться программой PdbCron означает уметь составить для нее расписание. Расписание PdbCron представляет собой текстовый документ, включающий один или несколько элементов расписания. Каждый элемент, в свою очередь, представляет собой описание момента (или моментов) времени, когда должна стартовать обработка, за которым следует список имен баз данных на сервере СУБД, подлежащих обработке, и, наконец, последовательность элементов обработки, задающих собственно шаги обработки, т. е. каждый элемент расписания задает триаду: "когда", "с какими БД", "что именно делать".

Описание момента времени задается в виде последовательности временных компонентов, разделенных пробелами, в одном из следующих форматов:

- месячном: `"M год месяц день_месяца час минута"`
- недельном: `"W год неделя день_недели час минута"`
- годовом: `"Y год день_года час минута"`

Другими словами, описание момента времени начинается с латинской буквы, определяющей его формат, за которой следует набор временных компонентов. Как и в программе Cron, каждый временной компонент может задаваться целым числом, диапазоном чисел, набором чисел или диапазонов через запятую, символом *, означающим "любое значение". При этом применяется естественная нумерация месяцев (с 1 по 12), дней в году, дней в месяце (начиная с 1) и дней в неделе (1 — понедельник, 2 — вторник и т. д.), а номер недели в году вычисляется по правилам стандарта ISO 8601.

Список имен БД задается в одной из следующих форм:

- последовательность имен БД, разделенных запятыми;

• условие, накладываемое на имя БД (datname) в формате SQL-команды SELECT в кавычках (например, "where datname like ‘srd%’").

Список элементов обработки задается в форме последовательности элементов, каждый из которых размещается на отдельной строке. В качестве элементов могут быть использованы:

• SQL-предложение [4];
• полное имя хранимой функции [3] в форме "имя_схемы.имя_функции" с параметрами вызова;
• команды PostgreSQL: BEGIN (начать транзакцию), COMMIT (завершить транзакцию);
• команды PdbCron: DUMP (выполнить асинхронный дамп БД) и WAIT (ожидание завершения всех запущенных дампов).

На рис. 1 приведены несколько примеров элементов расписания, иллюстрирующих основные возможности. Примеры содержат комментарии, начинающиеся с символа #.

Конфигурационные файлы

Все файлы с расширением cfg, находящиеся в каталоге программы PdbCron, рассматриваются ею как конфигурационные файлы, хранящие общие настройки (параметры) и расписание.

Общие настройки задаются в форме

```
ключевое_слово = значение
```

Каждая настройка размещается в отдельной строке. В таблице перечислены основные настройки и связанные с ними ключевые слова.

Ключевое слово ClockInterval позволяет определить период (в минутах), с которым PdbCron просматривает расписание. Если он превышает 1 мин, то моменты времени в расписании следует задавать таким образом, чтобы программа не пропускала их. Например, если значение параметра равно 5, то компонент "минута" в каждом элементе расписания должен быть задан или звездочкой, или с помощью целых чисел, кратных 5.

Расписание PdbCront должно начинаться с заголовка [Crontab] в отдельной строке. После заголовка должны быть размещены элементы расписания.

Если в каталоге программы имеется несколько конфигурационных файлов, то все они обрабатываются совершенно одинаково. В каждом из них может содержаться заголовок [Crontab] и часть расписания. Программа упорядочит конфигурационные файлы по имени и последовательно соединит все такие части в одно расписание.

Настройки PdbCron

Ключевое слово	Настройка	Значение по умолчанию
DbHost	IP-адрес (или имя) сервера PostgreSQL	127.0.0.1
DbPort	Номер порта сервера PostgreSQL	5432
MaxLogs	Максимальное число файлов журнала	4
MaxLogLen	Максимальная длина файла журнала, байт	10 000 000
MaxDumps	Максимальное число сохраняемых дампов для каждой БД	2
ClockInterval	Тактовый интервал, мин	1

```
# Выполнить 2 команды delete для БД srd1, srd2 и srd3 17 марта  2020
# года в 11:40.
M  2020  3  17  11  40     srd1,srd2,srd3
    delete from table1 where comment='tmp'
    delete from table2 where comment='tmp'

# Выполнять те же команды каждый год и каждый месяц по 5,10,11,12,13,14
# и 20  числам в 11:40.
M  *  *  5,10-14,20  11  40     srd1,srd2,srd3
    delete from table1 where comment='tmp'
    delete from table2 where comment='tmp'

# Выполнять хранимую функцию myscheme.function1 каждый год, каждую
# вторую неделю года по понедельникам и четвергам в 00:30.
W  *  */2  1,4  0  30     srd1,srd2,srd3
    myscheme.function1(100000.00,'test')

# Выполнять дамп всех БД с названием, начинающимся на "srd", каждый год
# с интервалом в 10 дней в  01:00. Дамп поместить в каталог  ./Dump
Y  *  */10  1  0     "where datname like 'srd%'"
    DUMP ./Dump
```

Рис. 1. Примеры элементов расписания PdbCron

Если конфигурационный файл содержит и общие настройки, и расписание, то общие настройки должны предшествовать расписанию. Например,

```
DbHost=192.168.1.24
MaxLogs=3
MaxDumps=2

[Crontab]
M * * */3 0 30 srd1,srd2,srd3
   myscheme.function1(100000.00, 'test')
   myscheme.function2()

W * */3 1,4 0 30 "where datname like 'srd%'"
INSERT INTO mytab1 (field1, field 2, ...) VALUES
(val1. Val2,...)
UPDATE mytab2 SET field1=value1, field2=value2,...
```

Логика работы программы PdbCron заключается в периодическом чтении всех конфигурационных файлов с интервалом в одну или несколько минут, проверке расписания и в выполнении всех действий, релевантных текущему моменту времени. В промежутке между проверками расписания программа находится в состоянии ожидания.

Обработка SQL-предложений

Все SQL-предложения обрабатываются последовательно, в том порядке, в котором они размещены в элементе расписания. Если при выполнении какого-либо предложения возникла ошибка (например, при выполнении предложения UPDATE произошло нарушение уникальности ключевого поля или при выполнении DELETE — разрушение связей с дочерними записями), то в журнал PdbCron будет внесена соответствующая запись, а обработка будет продолжена.

Предложение BEGIN позволяет объявить транзакцию с возможностью отката. Рассмотрим, например, следующий список SQL-предложений:

```
BEGIN
INSERT INTO mytab (field1, field2,...) VALUES (val1, val2,...)
UPDATE mytab SET field1=value1, field2=value2,...
DELETE FROM mytab WHERE field1=value1...
UPDATE mytab2 SET field1=value1, field2=value2,...
```

Если при выполнении предложения DELETE возникнет ошибка, то все изменения в БД, внесенные предшествующими двумя командами, будут отменены, в журнал PdbCron будет внесена запись об ошибке, а обработка элемента расписания будет прекращена. Если же после предложения INSERT поместить COMMIT, то изменения, внесенные этим предложением, будут сохранены (в отличие от изменений, внесенных предложением UPDATE):

```
BEGIN
INSERT INTO mytab (field1, field2,...) VALUES (val1, val2,...)
COMMIT
BEGIN
UPDATE mytab SET field1=value1, field2=value2,...
DELETE FROM mytab WHERE field1=value1...
UPDATE mytab2 SET field1=value1, field2=value2,...
```

Использование предложения SELECT в расписании PdbCron бессмысленно, так как у программы нет возможности воспользоваться его результатами. В случае необходимости предложение SELECT вместе с обработкой его результатов должны быть помещены в хранимую функцию СУБД PostgreSQL.

Вызов хранимых функций

Хранимые функции, обрабатываемые программой PdbCron, должны удовлетворять следующему условию — они должны возвращать значение типа record с двумя полями:

- типа integer (код завершения);
- типа varchar (диагностическое сообщение).

Другими словами, заголовок описания функции должен выглядеть так:

```
CREATE OR REPLACE FUNCTION myscheme.myfunc(..., OUT
rescode integer, OUT resmsg varchar)
RETURNS record
```

Вместо многоточия могут быть размещены описания произвольного числа входных параметров простых типов (integer, varchar, numeric). Если функция не удовлетворяет вышеуказанному требованию, то попытка вызова закончится аварийно, а в журнал PdbCron будет внесено сообщение о несоответствии формата возвращаемого значения требованиям программы.

Возвращаемые функцией данные обрабатываются следующим образом: если код завершения имеет отрицательное значение, то в журнал PdbCron вносится запись, содержащая диагностическое сообщение функции.

Рассмотрим следующий пример. Предположим, что система управления предприятием опирается на БД предприятия в СУБД PostgreSQL. Предположим также, что средства предприятия разбиты на несколько счетов, а в составе БД имеются таблицы ACCOUNT и TRANSFER для отслеживания состояния каждого счета и регистрации денежных проводок с одного счета на другой соответственно. Будем считать, что основными полями в таблице ACCOUNT являются ACCOUNT_ID (уникальный идентификатор счета) типа integer, ACCOUNT_BALANCE (текущий остаток на счете) типа numeric и ACCOUNT_START (начальная сумма на счете) типа numeric, а основными полями в таблице TRANSFER являются ACC_FROM (идентификатор счета-источника) типа integer, ACC_TO (идентификатор счета-приемника) типа integer и TR_SUM (сумма проводки) типа numeric.

На рис. 2 приведен иллюстративный пример кода хранимой функции, которая осуществляет контроль корректности счетов путем проверки соответствия данных в таблице ACCOUNT данным в таблице TRANSFER: текущий остаток каждого счета должен быть равен его начальному значению, увеличенному на сумму всех проводок на этот счет и уменьшенному на сумму всех проводок с этого счета.

Обращаем внимание, что заголовок функции соответствует требованиям программы PdbCron.

```
CREATE OR REPLACE FUNCTION myscheme.check_accounts(OUT rescode integer,
OUT resmsg varchar)
RETURNS record
LANGUAGE plpgsql
AS $function$
declare

AccId   integer;            --идентификатор обрабатываемого счета
AccCnt  integer;            --счетчик обработанных счетов
Balance numeric(12,2);      --остаток обрабатываемого счета
Start   numeric(12,2);      --начальная сумма на обрабатываемом счете
SumFrom numeric(12,2);      --всего поступило
SumTo   numeric(12,2);      --всего убыло
AccList varchar(1000);      --список идентификаторов 'проблемных' счетов

curTrans cursor for
 SELECT ACCOUNT_ID, ACCOUNT_BALANCE, ACCOUNT_START,
 -- всего поступило
 (SELECT SUM(TR_SUM) FROM TRANSFER TR WHERE TR.ACC_TO=AC.ACCOUNT_ID),
 -- всего убыло
 (SELECT SUM(TR_SUM) FROM TRANSFER TR WHERE TR.ACC_FROM=AC.ACCOUNT_ID)
 FROM ACCOUNT AC;

begin
    resmsg := ' '; rescode := 0; AccList :=''; AccCnt :=0;
    open curTrans;
    loop
      fetch curTrans
      into AccId, Balance, Start, SumTo, SumFrom;  --информация о счете
      exit when not found;
      if AccCnt >= 100 then exit; end if;
      if Balance != Start + SumTo - SumFrom then  --проверка  корректности
        if length(AccList)+length(AccId) < 1000 then
          AccList := AccList || AccId || ' ';   --формирование списка счетов
        end if;
      end if;
      AccCnt := AccCnt+1;
    end loop;
    close curTrans;

    if AccList == '' then                       -- формирование результата
       rescode := 0; resmsg := 'OK';
    else
       rescode := -1; resmsg := 'Проблемы со счетами: ' || AccList;
    end if;
end;
$function$
;
```

Рис. 2. Пример текста хранимой функции для использования в PdbCron

После циклического перебора и проверки всех счетов процедура возвращает нулевой код завершения (rescode), если все проверки прошли успешно, либо отрицательный код завершения и список номеров "неблагополучных" счетов в диагностическом сообщении (resmsg). В последнем случае диагностическое сообщение будет автоматически занесено в журнал PdbCron.

Предположим, что мы хотим использовать PdbCron для ежесуточного запуска этой функции в 03:00 с занесением отрицательного результата в журнал программы (будем считать, что всякая работа с БД в это время прекращается, т. е. можно не обременять пример командами блокировки). Соответствующий элемент расписания выглядит так:

```
M * * * 3 0 mybase
   myscheme.check_accounts()
```

Предполагается, что mybase — имя БД предприятия, а myscheme — имя схемы, в которой определена функция check_accounts.

Журнал

Так как программа PdbCron работает в режиме демона (без связи с монитором, клавиатурой и мышью), то единственным средством общения с оператором является ее журнал. В него заносится информация о неполадках, обнаруженных в процессе обработки, а также диагностические сообщения, возвращенные хранимыми процедурами.

Журнал представляет собой систему текстовых файлов, размещенных в подкаталоге Log каталога программы, с именами PdbCron.log, PdbCron1.log, PdbCron2.log и т. п. Новые сообщения всегда записываются в конец PdbCron.log. Когда он достигает максимально допустимого размера (общая настройка MaxLogLen), то все файлы переименовываются с возрастанием номера на 1, и создается новый пустой файл PdbCron.log. Если при этом число файлов превысит максимально разрешенное число (общая настройка MaxLogs), то самый "старый" из них стирается.

Дампы

Формирование дампов (резервных копий) баз данных инициируется с помощью предложения DUMP, которое имеет следующий формат:

```
DUMP [каталог_дампа[, каталог_журнала [, имя_журнала]]]
```

В этом предложении:

- "каталог_дампа" — полная спецификация каталога, в котором формируются дампы (значение по умолчанию ./Dump); при каждой обработке предложения Dump в этом каталоге будет создан подкаталог с именем, отражающим дату и время дампа (в формате YYYYMMDD_HHmm); в этом подкаталоге будут сформированы файлы дампов всех баз данных, указанных в элементе расписания; если число подкаталогов превысит максимально допустимое значение (общая настройка MaxDumps), то самый "старый" из подкаталогов будет удален со всем содержимым.
- "каталог_журнала" — полная спецификация каталога, в котором формируется журнал дампа (значение по умолчанию ./Log); в этот журнал заносится информация о неполадках, обнаруженных в процессе формирования дампов; журнал представляет собой систему текстовых файлов, размещенных в указанном каталоге; эти файлы формируются по тем же правилам, что и журнал PdbCron под управлением общих настроек MaxLogs и MaxLogLen;
- "имя_журнала" — общее базовое имя всех файлов журнала дампа (значение по умолчанию PdbCron); например, если задать значение MyDump, файлы журнала будут формироваться с именами MyDump.log, MyDump1.log, MyDump2.log и т. п.

Как видно из описания параметров предложения DUMP, предусмотрена возможность ведения собственных журналов дампов (в дополнение к журналу программы PdbCron). Это вызвано специфическим способом формирования записей в журнале дампа: они формируются на основе потока сообщений об ошибках от внешней программы pg_dump[6], которая используется для создания дампов БД. Если предложение DUMP задано без параметров, то эти сообщения будут направляться в единый журнал PdbCron, размещенный в подкаталоге Log каталога программы.

Рассмотрим следующий пример. Предположим, что на сервере СУБД PostgreSQL имеется множество баз данных, владельцы которых географически локализованы в различных временных поясах РФ. В целях безопасности владелец сервера хочет организовать периодическое ежедневное формирование дампов всех БД, причем создание дампа каждой БД должно начинаться после полуночи в соответствующей временной зоне. Предположим, что названия всех БД, владельцы которых локализованы во временной зоне NN по Гринвичу (GMT NN), начинаются с gmNN (например, БД московских владельцев могут иметь имена gm03n001, gm03n002, gm03n003 и т. п). Дампы БД каждой временной зоны должны формироваться в отдельном каталоге. Расписание программы PdbCron для решения этой задачи могло бы быть составлено следующим образом:

```
[Crontab]
M * * * 1 0 "where datname like 'gm02%'"
   DUMP ./Dump/GMT02 ./Dump PdbDump
M * * * 0 0 "where datname like 'gm03%'"
   DUMP ./Dump/GMT03 ./Dump PdbDump
M * * * 23 0 "where datname like 'gm04%'"
   DUMP ./Dump/GMT04 ./Dump PdbDump
        ...
M * * * 15 0 "where datname like 'gm12%'"
   DUMP ./Dump/GMT12 ./Dump PdbDump
```

Сформированная структура дампа приведена на рис. 3. Как видно из рисунка, все дампы размещены в подкаталоге Dump каталога программы. Для каждой временной зоны отведен отдельный подкаталог (от GMT02 до GMT12). В каждом таком каталоге происходит накопление дампов, сформированных в различные моменты времени (в данном случае предполагается, что параметр общей настройки MaxDumps равен 2 — поэтому в каталоге каждой зоны имеется только два подкаталога дампа). Собственно файлы дампов получают название от имени БД (например БД с именем gm03n002 соответствует файл дампа gm03n002.sql.gz).

В том же подкаталоге Dump формируется и отдельный журнал дампа (файлы PdbDump.log, PdbDump1.log, ... PdbDumpN.log), общий для всех временных зон.

Состав и структура программы

На рис. 4 приведена общая структура программы PdbCron. Как и для всякой программы на языке C++, сразу после запуска управление получает "главная"

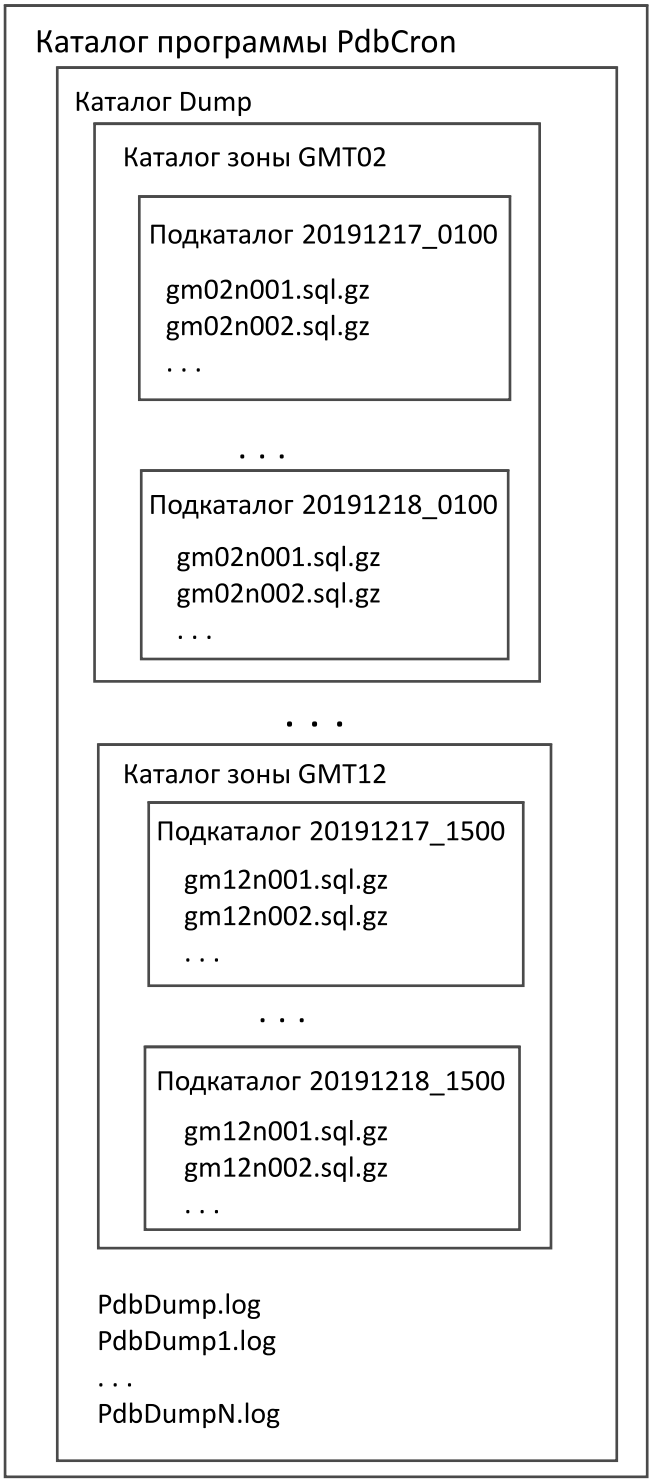

Рис. 3. Пример структуры каталогов дампа и журнала дампа

функция main. Она немедленно выполняет ряд действий, характерных для фоновых программ:

- переключение в режим "демона" с разрывом связей с монитором, клавиатурой и мышью;
- установка собственных обработчиков так называемых "сигналов" ОС Linux для потребного реагирования на экстренные ситуации (например, для создания записи в журнале о получении сигнала SIGTERM перед прекращением работы по команде оператора).

Кроме того, функция main выполняет ряд функций, специфичных для PdbCron, по управлению периодами активности и пассивности программы. Это управление сосредоточено в "вечном цикле" функции main, обеспечивающем обработку тактовых интервалов времени. В начале каждого тактового интервала времени программа "просыпается" и выполняет действия по обработке расписания, после чего управление возвращается в функцию main, которая запрашивает у системы текущее время в секундах (системный вызов ctime()) и определяет отрезок времени до начала следующего интервала, после чего переводит программу в пассивное состояние на этот отрезок времени (системный вызов sleep()). Например, если настройка ClockInterval равна 5 (300 с), а ctime(NULL)%300 = 110 (т. е. остаток от деления текущего времени на 300 равен 110), то функция main выполнит вызов sleep(190), что обеспечит выход на начальную секунду следующего тактового интервала после 190 с ожидания. Все остальные процедуры обеспечивают проверку и обработку расписания сразу же после активизации программы.

Обработчик конфигурации выполняет поиск и считывание всех конфигурационных файлов в лексикографическом порядке. Он же обеспечивает проверку всех фрагментов расписания и поиск элементов расписания, релевантных моменту начала тактового интервала. Все такие элементы последовательно передаются на обработку центральной процедуре PdbCron — обработчику элемента расписания.

Обработчик элемента расписания обеспечивает поиск и последовательное соединение со всеми БД, указанными в элементе расписания, и применение ко всем им одних и тех же действий, определенных списком элементов обработки. Для выполнения каждого действия вызывается одна из специализированных процедур-обработчиков.

Обработчик SQL-предложений и процедура вызова хранимых функций инициируют потребное действие в БД с использованием библиотеки libpq, обеспечивающей программный интерфейс к СУБД PostgreSQL. Разница между ними заключается главным образом в том, что первый заносит в журнал PdbCron лишь сообщения об ошибках от сервера СУБД, а вторая добавляет к ним еще и диагностические сообщения от хранимых функций.

Процедура формирования дампа обеспечивает вызов утилиты pg_dump для каждой базы данных, заданной в элементе расписания. Предварительно процедура создает новый подкаталог для дампа и, возможно, удаляет старые подкаталоги. Для каждого вызова pg_dump создается отдельный "сыновний" процесс с помощью системного вызова fork() [1]. Еще до передачи управления утилите pg_dump "сыновний" процесс, в свою очередь, создает "внучатый" процесс: вспомогательный процесс для специализированной утилиты PdbLogger, обеспечивающий запись сообщений об ошибках от утилиты pg_dump в журнал дампа по правилам программы PdbCron. Кроме того, программа обеспечивает соединение двух порожденных процессов с помощью механизма

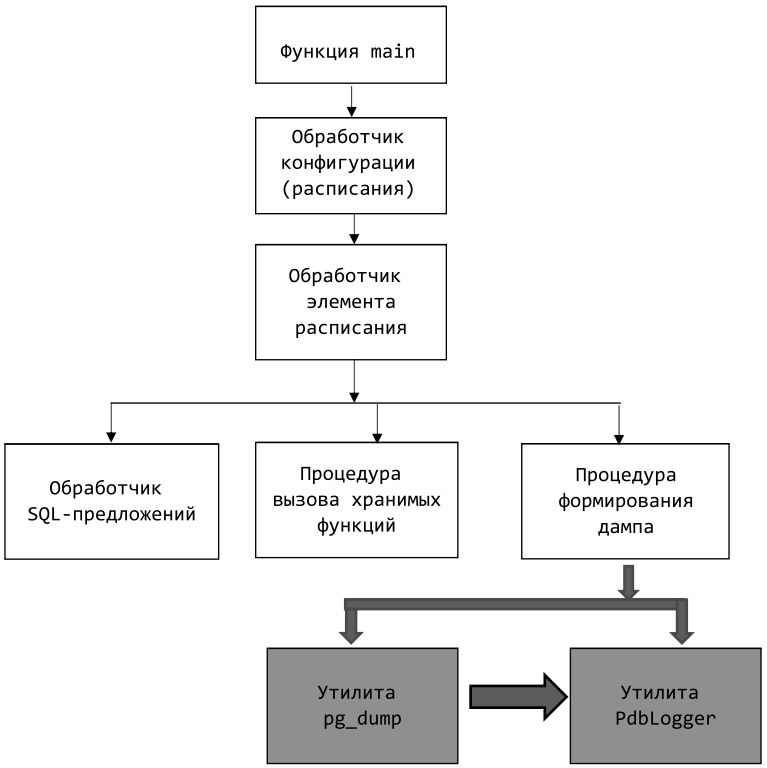

Рис. 4. Структура программы PdbCron

pipeline: стандартный вывод сообщений об ошибках (файловый дескриптор 2) процесса pg_dump связывается со стандартным вводом (файловый дескриптор 0) процесса PdbLogger с помощью системных вызовов pipe() и dup2() [1] (на рис. 4 это соединение отображено массивной фигурной стрелкой). Вся работа по связыванию процессов и перенаправлению файловых дескрипторов на pipeline сосредоточена в процедуре формирования дампа, которая работает как в главном процессе PdbCron, так и в "сыновнем", и во вспомогательном процессах благодаря внутренней логике системного вызова fork(). Сразу же после завершения работы утилиты pg_dump утилита PdbLogger заканчивает свою работу.

Установка и запуск программы

По умолчанию программа устанавливается в каталог /var/pdbcron. Установка выполняется с правами суперпользователя путем переписи всего каталога pdbcron с дистрибутивного носителя в каталог /var и последующего запуска скрипта PdbCronInstall.sh из этого каталога, который устанавливает правильные флаги доступа у исполняемых файлов и скриптов.

Ручной запуск программы осуществляется от имени суперпользователя с помощью скрипта PdbCronStart.sh, который обеспечивает проверку, не запущен ли уже соответствующий процесс, выбор каталога программы в качестве текущего каталога и собственно запуск:

```
#!/bin/sh
#
cd /var/pdbcron
ps h -CPdbCron | grep -q PdbCron
if [ $? -eq 0 ]
then
   echo "PdbCron is already running"
   exit 1
else
   echo "Starting PdbCron"
   ./PdbCron
fi
exit 0
```

Прекращение выполнения программы выполняется скриптом PdbCronStop.sh, который сначала определяет идентификатор соответствующего процесса, а затем "убивает" его командой kill:

```
#!/bin/sh
#
for pdbpid in $(ps h -CPdbCron -o pid)
do
   kill $pdbpid
done
exit 0
```

Автоматический запуск программы осуществляется или путем включения команды запуска в файл /etc/rc.local (если он есть) или по правилам так называемой "системы V": с использованием файла /etc/init.d и файлов /etc/rcN.d.

Все обращения к серверу СУБД программа выполняет от имени пользователя PostgreSQL. Предполагается, что этот пользователь имеет доступ ко всем БД. Пароль этого пользователя должен быть занесен в файл с именем .pgpass, подготовленный в защищенном каталоге /root по правилам утилиты pg_dump [6, 7]. Программа PdbCron считывает пароль из этого файла сразу после запуска. Если она не может определить пароль пользователя PostgreSQL, то заносит диагностическое сообщение в журнал и немедленно заканчивает свою работу.

Важное замечание. Согласно правилам утилиты pg_dump, пароль для доступа к БД хранится в файле pgpass в открытом виде. Несмотря на то, что этот файл принадлежит суперпользователю ОС, а право на чтение и запись имеет только его владелец (флаги доступа 0600 являются обязательным требованием pg_dump), суперпользователь ОС имеет доступ к паролю. Остается надеяться, что в следующих версиях pg_dump будет найден способ устранить эту брешь в защите.

Заключение

Хотя программа PdbCron предназначалась для применения во вполне конкретном проекте, были приложены все усилия к тому, чтобы особенности этого проекта никоим образом не отразились на ее свойствах и возможностях. Максимальная универсальность и возможность повторного использования рассматривались в числе основных приоритетов (в преддверии перевода сразу нескольких информационных систем на платформу Linux — PostgreSQL).

Программа PdbCron (как и программа PdbLogger) написана на языке C++ с использованием продуктов Postgres: библиотеки функций libpq и утилиты pg_dump. Суммарный объем исходного кода составляет около 110 Кбайт. Во время работы программа PdbCron занимает около 1,5 Мбайт оперативной памяти.

Список литературы

1. **Negus C.** Linux Bible. N. J. Wiley, 2015. 912 p.
2. **Таненбаум Э., Бос Х.** Современные операционные системы. СПб.: Питер СПб, 2019. 1120 с.
3. **Моргунов Е.** PostgreSQL. Основы языка SQL. СПб.: БХВ-Петербург, 2018. 336 с.
4. **Грофф Д., Вайнберг П., Оппель Э.** SQL. Полное руководство. Киев: Диалектика, 2019, 960 с.
5. **Блог** компании Postgres Professional. Что умеет планировщик заданий в Postgres Pro, 2017. URL:https://habr.com/ru/company/postgrespro/blog/335798/
6. **PostgreSQL 9.6.16** Documentation .pg_dump, 2019. URL: https://www.postgresql.org/docs/9.6/app-pgdump.html
7. **PostgreSQL 9.3.25** Documentation. Chapter 31. libpq — C Library, The Password File, 2019. URL: https://www.postgresql.org/docs/9.3/libpq-pgpass.html

Scheduled Data Processing Service for the PostgreSQL DBMS in Linux Environment

R. E. Asratian, rea@ipu.ru, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation

Corresponding author:

Asratian Ruben E., Leading Researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation
E-mail: rea@ipu.ru

Received on February 11, 2020
Accepted on February 14, 2020

The basics of the organization and operation of the background program PdbCron, designed to perform periodic scheduled data processing in the PostgreSQL DBMS (cleaning temporary or outdated data, checking the correctness or backup data) on its own schedule in automatic mode in the operating environment of Linux are considered. The approach is entirely founded on free distributed Postgres software (including libpq function library and utility pg_dump). Likewise a well-known program Cron the program PdbCron is a permanently active "daemon" in Linux environment, but is entirely focused on databases processing instead of tasks scheduling in the system. Unlike the Cron, the program uses different formats of schedule items based not only on the month format (year — month — day — hour — minute), but on the week format (year — week — day — hour — minute) or short year format (year — day of year — hour — minute). The number of a week in the year is determined according to ISO 8601 standard. Every schedule item can be associated with an arbitrary list of processing steps applied to arbitrary subset of databases on a Postgres server (including an SQL-statement execution, a stored function call or dump generation). The structure of PdbCron and the functions of its main components are described. Although the program was intended for use in a very specific project, its authors tried to develop it so that the features of this project in no way affect its properties and capabilities. Maximum versatility and the capability of reuse were considered among the main priorities.

Keywords: *information systems, data bases, PostgreSQL DBMS, OS Linux, operation schedule*

For citation:

Asratian R. E. Scheduled Data Processing Service for the PostgreSQL DBMS in Linux Environment, *Programmnaya Ingeneria*, 2020, vol. 11, no. 2, pp. 115—122

DOI: 10.17587/prin.11.115-122

References

1. **Negus C.** *Linux Bible*, N.J., Wiley, 2015, 912 p.
2. **Tanenbaum A., Boss H.** *Modern Operating Systems*, Saint-Petersburg, Piter, 2019. 1120 p. (in Russian).
3. **Morgunov E.** *PostgreSQL. SQL language principles,* Saint-Petersburg, BHV-Peterburg, 2018, 336 p. (in Russian).
4. **Groff J., Weinberg P., Oppel A.** *SQL. The Complete Reference*, Kiev, Dialektika, 2019, 960 p. (in Russian).
5. **The blog of Postgres Professional**. What the task scheduler can do in Postgres Pro, 2017, available at: https://habr.com/ru/company/postgrespro/blog/335798/
6. **PostgreSQL 9.6.16** Documentation .pg_dump, 2019, available at https://www.postgresql.org/docs/9.6/app-pgdump.html
7. **PostgreSQL 9.3.25** Documentation. Chapter 31. libpq — C Library, The Password File, 2019, available at https://www.postgresql.org/docs/9.3/libpq-pgpass.html

УДК 004.056.5 DOI: 10.17587/prin.11.123-128

Е. С. Васева, канд. пед. наук, доц., e-s-vaseva@mail.ru,
Н. В. Бужинская, канд. пед. наук, доц., nadezhdabuzh@gmail.com,
М. С. Шушпанов, магистрант, max.shushpanov@gmail.com, Нижнетагильский государственный социально-педагогический институт (филиал) ФГАОУ ВО "Российский государственный профессионально-педагогический университет", Нижний Тагил

К разработке информационной системы автоматизации процессов защищенного хранения и передачи данных

Представлен принятый авторами относительно простой подход к разработке системы, предназначенной для защиты текстовых данных посредством шифрования. Выполняется процедура отбора средств разработки, обсуждаются механизмы функционирования этой системы. При применении предложенной системы пользователь может выбрать алгоритм шифрования, выполнить шифрование, получить ключ в виде текста или QR-кода, расшифровать данные, записать зашифрованный текст в базу данных на удаленном сервере.

Ключевые слова: *шифр, шифрование, расшифровка, алгоритм шифрования, AES, SHA-1, RSA, информационная система, приложение, база данных*

Введение

Проблема защиты данных в современном мире активного развития информационно-коммуникационных технологий является очень актуальной. Наличие систем защиты информации в государственных и крупных коммерческих организациях обусловлено необходимостью выполнения федеральных законов, эти системы ориентированы на выполнение стандартов и, как правило, имеют комплексное решение. В деятельности малого и среднего бизнеса, в отличие от крупных предприятий, проблема внедрения и развития систем защиты информации не является приоритетной. На первое место ставятся задачи, связанные с выпуском продукции, оказанием услуг или куплей и продажей товаров. Вместе с тем существуют риски, связанные с обеспечением безопасности хранения и передачи данных в деятельности малого и среднего бизнеса. Небольшие компании ограничены в финансовых ресурсах, ориентированы на применение облачных технологий, как правило, не имеют в штатном расписании специалиста, ответственного за обеспечение информационной безопасности. Поэтому весьма востребованным является создание простой в эксплуатации, не требующей значительных ресурсозатрат при внедрении системы автоматизации процессов защищенного хранения и передачи данных, которая адекватна потребностям и возможностям предприятий малого и среднего бизнеса [1]. Внедрение на таком предприятии системы автоматизации процессов шифрования данных поможет сделать информацию более защищенной и избежать сбоев в осуществлении основной деятельности [2—5].

Постановка задачи

Разработка подобной системы ставит перед программистом две важные задачи. Во-первых, программное обеспечение (ПО) должно быть максимально дружелюбным к пользователю. Чем проще оно устроено, тем больше людей сможет им воспользоваться, а предприятия без лишних затрат на подготовку персонала смогут внедрить это ПО в свою корпоративную среду. Во-вторых, следует учитывать отказоустойчивость оборудования и предоставить пользователю возможность сохранения необходимых данных для расшифровки. В противном случае все может обернуться необратимой потерей информации, расшифровка которой может быть просто невозможной в силу защищенности алгоритма.

В процессе разработки системы особое внимание следует уделить целям функционирования и возможностям создаваемой системы.

Анализ аналогичных приложений, таких как Encrypto [6], Silver Key [7], Steganos Safe [8] и др., позволяет сделать вывод, что рассматриваемые программные решения имеют один схожий недостаток — отсутствие гибкости выбора алгоритма шифрования. Выбор алгоритма шифрования дает возможность реализовать многоуровневую защиту, например, текст можно зашифровать симметричным алгоритмом шифрования, а ключ — ассиметричным. Большинство рассмотренных приложений не бесплатны и имеют лишь пробный срок, в течение которого существует возможность ознакомиться с приложением.

На наш взгляд, информационная система для защищенного хранения данных должна предоставлять следующие функциональные возможности:

- надежное шифрование текстовых данных;
- расшифровку текстовых данных;
- сохранение шифруемых данных в облачном хранилище;
- выбор приоритета сохраняемых данных;
- поиск нужного контента среди сохраненных данных;
- гибкий выбор алгоритмов и масштабируемость;
- выбор языка приложения;
- настройку всех важных аспектов разрабатываемой системы;

- допустимость загрузки текста;
- получение ключа надежными средствами;
- максимальную независимость приложения от дополнительных манипуляций, кроме установки.

По окончанию процесса шифрования пользователь получает зашифрованный текст и ключ к нему, что дает возможности сохранения контента несколькими удобными способами и передачи средствами приложения.

Таким образом, построение системы, относительно простой для эксплуатации и малозатратной при ее внедрении, удовлетворяющей перечисленным выше требованиям, являлось предметом прикладных исследований и инженерных работ, результаты которых представлены в настоящей статье.

Разработка пользовательского интерфейса

Разрабатываемая система будет состоять из базы данных, куда будут записываться зашифрованные данные, и приложения, определяющего функциональные возможности системы. При необходимости пользователь может отказаться от записи зашифрованных данных в удаленную базу данных и сохранить их на текущем компьютере. Сохранение информации в удаленной базе данных позволяет организовать большую сохранность их и при необходимости обеспечить совместную работу. Для хранения данных будет использоваться система управления базами данных MySQL.

В основе приложения лежит язык C#, а целевой платформой является Windows. Для данного сочетания существует специальный интерфейс программирования приложений Windows Forms (разработанный Microsoft), полностью поддерживающий интерфейс прикладного программирования Windows (Windows API) [9]. Элементы в окне приложения поделены на две части и визуально разделены разным фоном. В левой части расположены элементы управления, которые не зависят от выбранного действия. В правой части элементы изменяются в зависимости от выбранного действия (рис. 1).

Приложение представляет собой набор вкладок, каждая из которых скрыта для пользователя, а переход между ними осуществляется средствами интерфейса пользователя. Две вкладки предназначены для шифрования и расшифровки текста. В обоих случаях текст может быть набран с клавиатуры или загружен из файла (рис. 2).

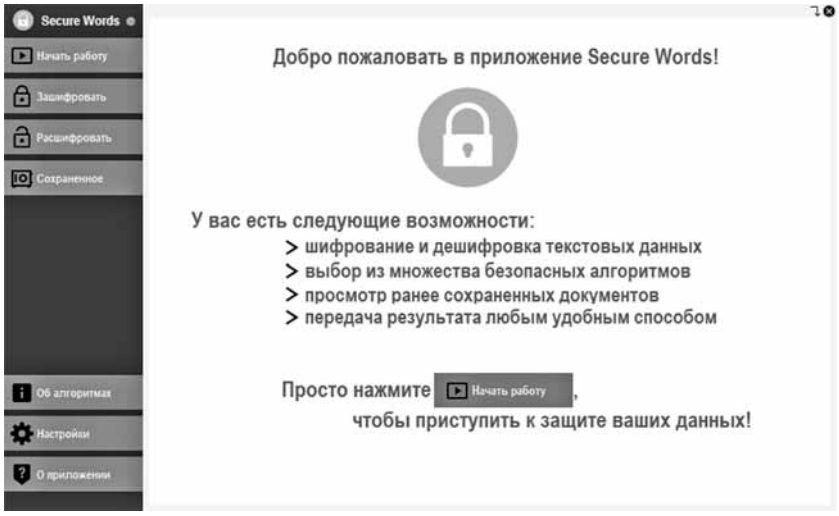

Рис. 1. Вид интерфейса приложения

Рис. 2. Вид активного меню расшифровки данных

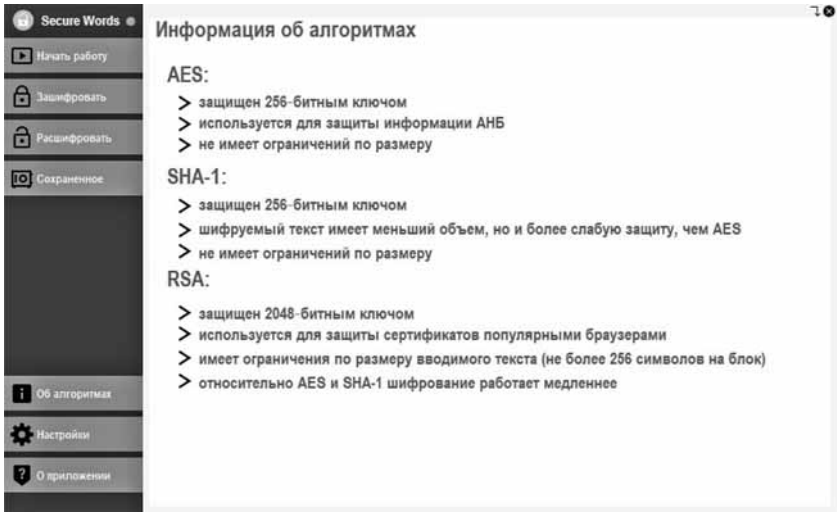

Рис. 3. Вид меню с описанием алгоритмов

Особенности алгоритмов шифрования также вынесены в отдельную форму, по пунктам расписаны ключевые особенности каждой методики, а также предусмотрен переход в браузер по ссылке на описание данного метода шифрования, чтобы пользователь мог подробнее с ними ознакомиться и сделать осознанный выбор при защите своих текстовых данных (рис. 3).

Для ранее сохраненных данных предусмотрено отдельное меню, где выходные данные представлены в виде таблицы с указанием зашифрованного текста, алгоритма его шифрования и даты (рис. 4). В целях безопасности ключ не сохраняется в удаленной базе данных, его хранение осуществляется любым удобным пользователю способом. Двойной клик по любому полю таблицы перенаправит пользователя в меню расшифровки, где уже будет представлен зашифрованный текст из выбранной строки, и останется лишь предоставить ключ, с помощью которого можно будет получить изначальный текст.

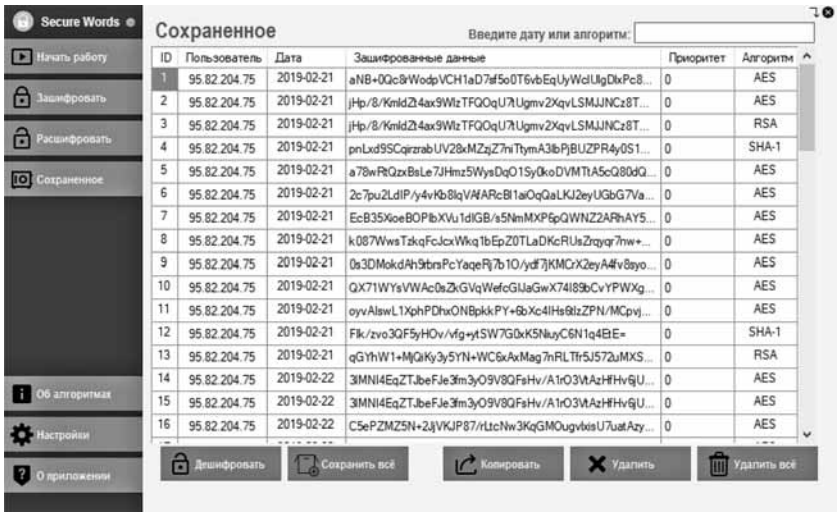

Рис. 4. Организация хранения сохраненных данных

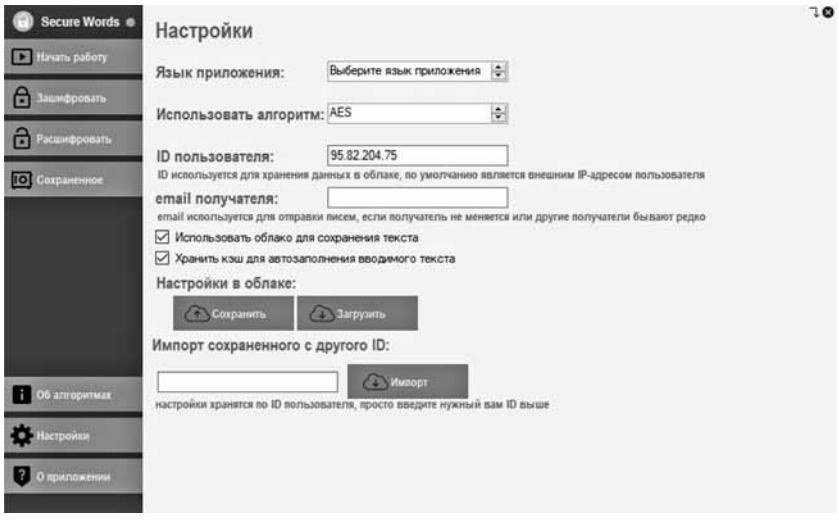

Рис. 5. Пример реализации окна настроек

Пользователь имеет доступ к окну настроек, в котором может задать удобные ему параметры использования приложения (рис. 5). Можно задать почту по умолчанию, именно на нее будут отправляться все письма в будущем. Существует возможность отключить облачное хранилище, чтобы приложение не тратило время на выход в Интернет. Для удобства предусмотрена возможность загрузки своих настроек в облако с последующей загрузкой из другого места нахождения. Также предусмотрена загрузка зашифрованных текстовых данных с другого профиля. Поскольку в базе данных не хранятся ключи, данный способ хранения полностью безопасен.

Разработка механизмов функционирования системы

Для осуществления процессов шифрования потребуется внедрение методов для реализации алгоритмов шифрования и расшифровки.

Внутри кода программы алгоритм AES реализуется на основе стандартного пространства имен System.Security.Cryptography, в котором существует класс RijndaelManaged. На основе данного класса осуществляется процесс шифрования текста по алгоритму AES, при этом сторонние решения не задействуются.

Для внедрения алгоритма SHA-1 используется класс PasswordDeriveBytes из пространства имен System.Security.Cryptography. При этом аргументами выступают ключ, изначальный текст, выбранный алгоритм шифрования (в нашем случае SHA-1) и число итераций шифрования. В нашем случае используется всего две итерации. Это позволяет ускорить работу приложения, при этом создается достаточно устойчивый хеш и ключ к нему.

В коде разрабатываемой программы алгоритм RSA реализуется использованием стандартного класса RSACryptoServiceProvider из пространства имен System.Security.Cryptography. Поскольку передачи данных между разными клиентами приложения не предусмотрено, открытый ключ не используется (хотя и создается). Следовательно, пользователь получает закрытый ключ, с помощью него и осуществляется расшифровка. Однако имеется ограничение на входной текст, его размер не должен превышать 256 символов, иначе приложение выдаст исключение, которое также учтено в коде. Таким образом, данный алгоритм не может использоваться для шифрования больших объемов данных, если не реализована разбивка на разные блоки.

Многопоточность позволяет выполнять вычисления, занимающие

Рис. 6. Пример ошибки подключения

определенное время, не прерывая работы приложения. В данном случае приложение при запуске параллельно пытается выполнить подключение к базе данных, в случае невозможности действия приложение переведет пользователя в отдельное окно и предложит подключиться повторно или продолжить работу без соединения с базой данных (рис. 6). При нажатии любой из опций пользователь вернется в предыдущее окно, где будут сохранены ранее внесенные изменения. Таким образом, пользователь никак не отвлекается на всплывающие окна и не ждет подключения, приложение запускается и работает в обычном режиме, но с ограничением функциональности, связанной с базой данных.

Многопоточность также используется при отправке почты. Для этого требуется открыть соединения, сформировать письмо и дождаться от протокола SMTP положительного ответа. Если выполнять напрямую эти действия, пользователь может столкнуться с зависанием приложения на время выполнения всех указанных действий, с многопоточностью же они будут выполняться параллельно.

Для работы с многопоточностью нужно обратиться к директиве System.Threading.Tasks, и желаемый метод должен объявляться с модификатором async. Следовательно, данный метод будет выполняться параллельно работе основного потока приложения.

Переключение языка может быть важно для англоговорящих работников, либо для тех, кто предпочитает пользоваться программным обеспечением на английском языке. Дополнительно может потребоваться локализация в зависимости от будущих направлений бизнеса, потому в коде предусмотрена масштабируемость выбора языка.

Для этого был создан класс LocalisedStrings, внутри которого находятся переменные, содержащие текст каждой локализации. Условно локации поделены по идентификаторам: 0 — русская, 1 — английская, дополнительные константы могут использоваться для новых локализаций. Внедрение в основной код происходит через инициализацию переменной типа LocalisedStrings и индекса локализации при запуске программы. Каждый раз происходит обращение к нужной переменной, и в зависимости от выбранного языка на экран выводится текст.

Кеширование позволяет сохранять некоторые действия пользователя, для повторного их использования в будущем и отсутствия необходимости вводить данные повторно. Например, текст окон шифрования и расшифровки сохраняется при закрытии приложения, в будущем пользователь может открыть приложение и продолжить с того места, на котором он остановился. При этом ключи никогда не сохраняются и требуются только в момент использования приложения.

Ключ может быть получен в виде кода быстрого отклика (QR-кода). Генерация QR-кодов осуществляется с помощью библиотеки ZXing. Ее внедрение в код происходит начиная с директивы using ZXing. Далее необходимо создать внутри формы объект, в котором будет содержаться изображение, обычно это пустой pictureBox. Нужно инициализировать переменную класса QRCodeWriter, определить 2D-матрицу битов для построения изображения методом encode в переменную класса BitMatrix, по-

сле чего записать в нее QR-код на основе входящих текстовых данных методом Write. Таким образом можно вывести на экран сгенерированный QR-код, который пользователь может прочитать любым поддерживаемым устройством или сохранить его для последующего чтения средствами разрабатываемого приложения. Чтение QR-кода происходит с помощью объекта класса BarcodeReader, от которого обращаемся к методу Decode, предварительно загрузив из файла, выбранного пользователем внутри файловой системы. Результат чтения QR-кода преобразуется в текст методом ToString и передается в окно ввода ключа, после чего пользователь может с помощью полученных данных расшифровать нужный ему зашифрованный текст.

Заключение

Разработанная система позволит пользователю без дополнительного обучения и финансовых затрат получить текстовые данные в защищенном от злоумышленников виде. При наличии ключа шифрования эти данные могут быть без труда расшифрованы и предоставлены в виде изначального варианта текста. Программа может быть использована в деятельности малого бизнеса, небольших организаций, индивидуальных предпринимателей и отдельных пользователей для дополнительной защиты конфиденциальной информации, представленной в текстовом виде, и при необходимости передачи.

По результатам исследования и разработки системы авторами было получено свидетельство Роспатент о государственной регистрации программы для ЭВМ "Автоматизированная система защищенного хранения и передачи данных" [10].

Список литературы

1. **Secure** News. Защита информации в небольших компаниях: приоритет или "дело десятое"? URL: https://securenews.ru/small_business/
2. **Галатенко В. А.** Кибербезопасность в здравоохранении // Программная инженерия. 2016. Т. 7, № 3. С. 117—125.
3. **Згоба А. И., Маркелов Д. В., Смирнов П. И.** Кибербезопасность: угрозы, вызовы, решения // Вопросы кибербезопасности. 2014. № 5. С. 30—38.
4. **Подуфалов Н. Д., Конявский В. А.** Защита электронного документооборота с помощью криптографических методов // Информатизация и связь. 2012. № 2. С. 57—60.
5. **Шниперов А. Н., Чистяков А. П.** Способ и информационная система для конфиденциального обмена информацией в открытых компьютерных сетях // Программная инженерия. 2017. Т. 8, № 8. С. 359—368.
6. **Encrypto.** URL: https://macpaw.com/encrypto.
7. **Silverkey**. URL: https://www.kryptel.com/products/silverkey.php
8. **Steganos.** URL: https://www.steganos.com/en/products/steganos-safe-21
9. **Скит Д.** C# для профессионалов: тонкости программирования. М.: Вильямс, 2014. 608 с.
10. **Свид. 2019662243** Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Автоматизированная система защищенного хранения и передачи данных / Бужинская Н. В., Шушпанов М. С., Васева Е. С.; заявитель и правообладатель: ФГАОУ ВО "РГППУ" (RU). № 2019661000, заявл. 05.09.2019, опубл. 19.09.2019, Реестр программ для ЭВМ. 1 с.

Towards the Development of an Information System for the Automation of Secure Storage and Data Transfer

E. S. Vaseva, e-s-vaseva@mail.ru, **N. V. Buzhinskaya**, nadezhdabuzh@gmail.com, **M. S. Shushpanov**, max.shushpanov@gmail.com, Nizhny Tagil state socio-pedagogical Institute (branch) of Federal State Autonomous educational institution Russian State Vocational Pedagogical University, Nizhny Tagil, 622031, Russian Federation

Corresponding author:

Vaseva Elena S., Associate Professor, Nizhny Tagil State Socio-Pedagogical Institute (branch) of Federal State Autonomous Educational Institution Russian State Vocational Pedagogical University, Nizhny Tagil, 622031, Russian Federation
E-mail: e-s-vaseva@mail.ru

Received on January 24, 2020
Accepted on February 28, 2020

Need for data encryption process automation system implementation at the operations is updated. Such system will help to make the information more protected and to avoid failures in company operating activities. Development strategy is considered for the system intended for alphanumeric data protection by means of encryption. Development kits are selected and the information system functioning mechanisms are discussed. The proposed system consists of an application realized through C# programming language means and a data base created in the MySQL data base control system.

Certain demands were placed on the system and taken into account in the course of development; they include user-friendly interface, reliable encryption of alphanumeric data, decryption of alphanumeric data, encrypted data storage in the Cloud Storage, the stored data priority selection option, search for required content among the saved data, flexible choice of encryption algorithm (AES, SHA-1, RSA); scalability; application language selection option; setting of all important aspects of the system under development; convenient text loading and key obtaining with reliable tools; key format selection option, automatic sending of an encrypted text, plain text, key or QR-code to email addresses, maximum application independence from additional operations except for installation.

The developed system will allow the user without any additional training or financial costs to receive alphanumerical data in the form protected from intruders. Upon completion of the encryption process the user will receive an encrypted text and a key to it with an option of saving it by several convenient ways and transferring it using the application tools.

The software may be used by small businesses, small companies, individual entrepreneurs and individual users for additional protection of confidential information represented in an alphanumerical form.

Keywords: *code, encryption, de-encryption, encryption algorithm, AES, SHA-1, RSA, information system, application, data base*

For citation:

Vaseva E. S., Buzhinskaya N. V., Shushpanov M. S. Towards the Development of an Information System for the Automation of Secure Storage and Data Transfer, *Programmnaya Ingeneria,* 2020, vol. 11, no. 2, pp. 123—128.

DOI: 10.17587/prin.11.123-128

References

1. **Secure News.** Information security in small companies: priority or "case ten"? available at: https://securenews.ru/small_business/ (in Russian).
2. **Galatenko V. A.** Cybersecurity in Healthcare, *Programmnaya Ingeneria*, 2016, vol. 7, no. 3, pp. 117—125 (in Russian).
3. **Zgoba A. I., Markelov D. V., Smirnov P. I.** Cybersecurity: threats, challenges, solutions, *Voprosy kiberbezopasnosti*, 2014, no 5, pp. 30—38 (in Russian).
4. **Podufalov N. D., Konyavskiy V. A.** Protection of electronic document management using cryptographic methods, *Informatizatsiya i svyaz'*, 2012, no. 2, pp. 57—60 (in Russian).
5. **Shniperov A. N., Chistyakov A. P.** The Method and Information System for the Exchange of Confidential Information in Open Computer Networks, *Programmnaya Ingeneria,* 2017, vol. 8, no. 8, pp. 359—368 (in Russian).
6. **Encrypto**, available at: https://macpaw.com/encrypto
7. **Silverkey**, available at: https://www.kryptel.com/products/silverkey.php
8. **Steganos**, available at: https://www.steganos.com/en/products/steganos-safe-21
9. **Skit D.** C# for professionals: the intricacies of programming. Moscow, Wil'yams, 2014, 608 p. (in Russian).
10. **Buzhinskaya N. V., Shushpanov M. S., Vaseva E. S.** Certificate 2019662243 Russian Federation. Certificate of official computer program registration. Avtomatizirovannaya sistema zashchishchennogo khraneniya i peredachi dannykh (Automated system for secure storage and data transfer) / Buzhinskaya N. V., Shushpanov M. S., Vaseva E. S.; applicant and rightholder: Federal State Autonomous educational institution "Russian state vocational pedagogical University", no 2019661000, application 05.09.2019, published 19.09.2019, Catalog of Computer Programs of Rospatent. 1 p. (in Russian).

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 13.02.2020 г. Подписано в печать 24.03.2020 г. Формат 60×88 1/8. Заказ PI220
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

Рисунки к статье Д. Д. Руховича

«ОЦЕНКА АБСОЛЮТНОГО МАСШТАБА В МОНОКУЛЯРНЫХ SLAM-СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ СИНТЕТИЧЕСКИХ ДАННЫХ»

Рис. 1. Примеры реальных изображений из набора KITTI (верхний ряд) и из симулятора CARLA (нижние три ряда) при различных времени дня и погодных условиях

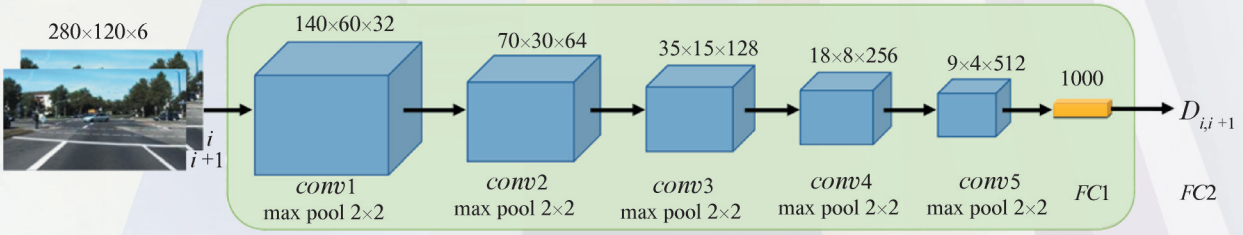

Рис. 2. Базовая архитектура сверточной сети (CNN), используемая во всех экспериментах

Рисунки к статье **Д. Д. Руховича**

«ОЦЕНКА АБСОЛЮТНОГО МАСШТАБА В МОНОКУЛЯРНЫХ SLAM-СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ СИНТЕТИЧЕСКИХ ДАННЫХ»

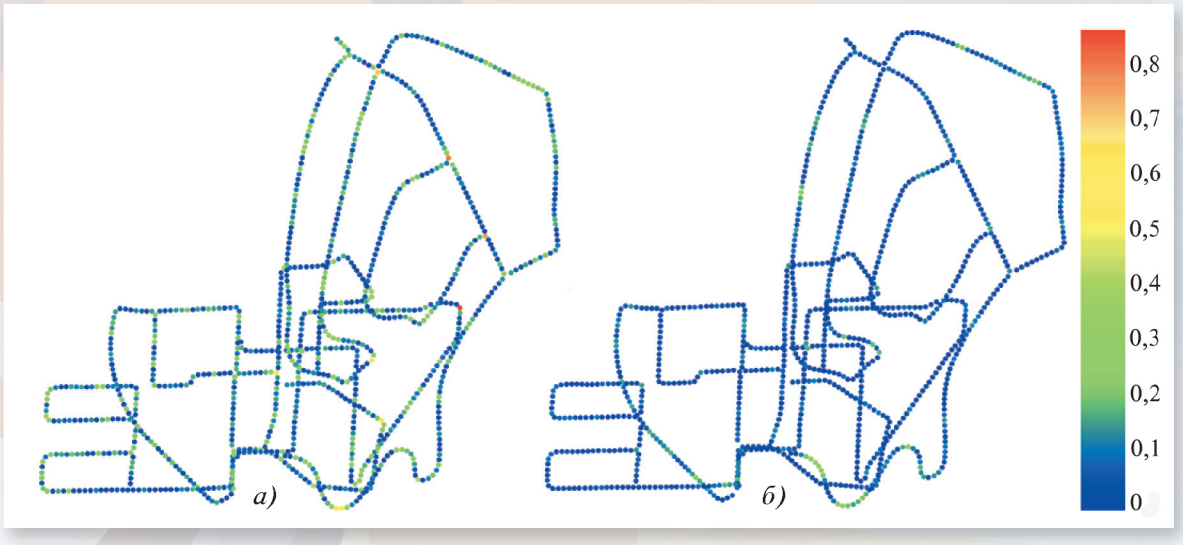

Рис. 5. Траектории набора KITTI, окрашенные в зависимости от уровня ошибки оценки абсолютного расстояния: *а* – базовый метод [5]; *б* – предложенный в данной работе метод; единица цветовой шкалы – метр

Рис. 6. Примеры пар соседних кадров тестовых видеопоследовательностей набора KITTI, дающих наибольшую ошибку

ISSN 2220-3397. Программная инженерия. 2020. Том 11. № 2. 65–128. Подписной индекс по Объединенному каталогу "Пресса России" – 22765