

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ЗАДАЧИ ПОИСКА СЛОЖНЫХ РИГИДНЫХ ОБЪЕКТОВ В СИСТЕМЕ ОБРАБОТКИ КОСМИЧЕСКИХ СНИМКОВ «ПС НСКИД»

Лебедев А.С.¹, Фраленко В.П.², Чэн Г.С.³, Чжан Г.Л.⁴

¹ФГБУН «Институт системного анализа» Российской академии наук, лаборатория 11-1 «Динамика макросистем» (117312, г. Москва, проспект 60-летия Октября, 9), e-mail: tementy@gmail.com

²ФГБУН «Институт программных систем им. А.К. Айламазяна» Российской академии наук, Исследовательский центр мультипроцессорных систем (152021, Ярославская обл., Переславский р-н, с. Вёськово, ул. Петра Первого, 4а), e-mail: alarmod@pereslavl.ru

³ООО «Ханчжоуская компания электронных технологий “AIVISI”» (311258, КНР, провинция Чжэцзян, г. Ханчжоу, р-н Сяошань, восточный вход культурно-спортивного центра городской управы административного центра Шеньян, 4-й этаж), e-mail: chengx@ketuoda.com

⁴Харбинский политехнический университет (150001, КНР, провинция Хэйлунцзян, г. Харбин, р-н Наньган, ул. Сылин, 11), e-mail: 0451_zgl@163.com

Работа посвящена российско-китайскому опыту сотрудничества при решении задачи обнаружения сложных ригидных (твердотельных) объектов на изображениях дистанционного зондирования Земли. Проблема поиска целевых объектов разбивается на две взаимосвязанные задачи: поиск зон интереса с помощью алгоритмов спектрографической «закраски» и распознавание выявленных зон сверточной нейронной сетью с визуализацией результатов. Приведены декларативные описания схем решаемых задач. Раскрыты особенности функционирования программных модулей. Применяется ряд инструментальных средств, позволяющих задействовать высокопроизводительные вычислительные устройства, в том числе графические ускорители. Разработанное программное обеспечение позволило осуществить обработку сверхбольших изображений дистанционного зондирования Земли в конвейерно-параллельном режиме с близким к линейному ускорению. Могут быть использованы различные виды параллелизма, такие как параллелизм независимых ветвей; множества объектов или данных; параллелизм смежных операций. Представлен специальный графический интерфейс для эффективного использования компонентов программно-инструментальной системы, позволяющий упростить решение прикладных задач. Интерфейс реализует поддержку функций универсальной моделирующей среды, обеспечивает задание необходимого набора вычислительных модулей, выбор их настроек; конфигурирование каналов связи; запуск и остановку распределенных вычислений; сохранение и визуализацию результатов.

Ключевые слова: сотрудничество, дистанционное зондирование, изображение, ригидный объект, параллелизм.

EXPERIMENTAL STUDY OF COMPLEX RIGID OBJECTS SEARCH PROBLEM IN THE SATELLITE IMAGES PROCESSING SYSTEM “PS NSKID”

Lebedev A.S.¹, Fralenko V.P.², Chen G.X.³, Zhang G.L.⁴

¹Institute for Systems Analysis of the Russian Academy of Sciences, Laboratory 11-1 “Dynamics of Macrosystem” (117312, Moscow, pr. 60-letiya Oktyabrya, 9), e-mail: tementy@gmail.com

²Ailamazyan Program Systems Institute of the Russian Academy of Sciences, Research Center for Multiprocessor Systems (152021, Yaroslavl region, Pereslavl area, Peter First st., 4a), e-mail: alarmod@pereslavl.ru

³Hangzhou Aivisi Electronic Co., Ltd. (311258, China, Zhejiang, Hangzhou, Xiaoshan District, East Side of Cultural and Sports Center of Wenyan Town Government, 4th Floor), chengx@ketuoda.com

⁴Harbin Institute of Technology (150001, China, Heilongjiang, Harbin, Nangang District, Siling Street, 11), 0451_zgl@163.com

The paper reflects the effect of Russia and China collaboration for solving the problem of detection of complex rigid objects on remote sensing images. The problem of detection of target objects is decomposed into two closely related tasks: the first is to choose regions of interest using spectrographic «shading» algorithms, the second is recognition of found zones using convolution neural networks machinery with visualization of the result. Declarative definitions of tasks being solved are included. All the peculiarities of software modules are described. The special set of tools is used to expose compute capabilities of high performance parallel hardware including graphics processing units. The software developed under research allows to process extra-large remote sensing images in pipelined parallel manner with almost linear speedup. Different levels of parallelism are covered: parallelism of independent branches; sets of objects or data elements; parallelism of adjacent operations.

Specifically designed graphical user interface helps to use the components of the software effectively. It implements functions of universal modelling environment, provides the ability to define the set of necessary computational modules with their parameters, configure communication channels, start and stop distributed computations, save and visualize results.

Keywords: collaboration, remote sensing, image, rigid object, parallelism.

Решение задачи автоматического обнаружения целевых объектов на сверхбольших снимках дистанционного зондирования Земли (ДЗЗ) продиктовано актуальностью проведения совместных исследований, инициированных Соглашением между Федеральным государственным бюджетным учреждением науки Институтом системного анализа Российской академии наук и Ханчжоуской компанией электронных технологий AIVISI о совместной инициативной работе (от 10 мая 2014 года). Исследования проводятся в рамках развития трансфера технологий между Российской Федерацией и Китайской Народной Республикой.

В широкомасштабных исследованиях приняли непосредственное участие сотрудники Ханчжоуской компании электронных технологий AIVISI и Харбинского политехнического университета, которые на двух совместных встречах, проведенных в рамках Международной ярмарки высоких технологий CHINA HI-TECH FAIR (г. Шеньчжень, Китай, 16-21 ноября 2012 г.) и Инновационного форума «Пуцзян» (г. Шанхай, Китай, 22-29 октября 2014 г.), дали реальные предложения по постановкам задач и методам их решения.

Среди возможных целевых объектов, являющихся объектами поиска на снимках ДЗЗ, большой интерес вызывают так называемые ригидные объекты, т.е. твердотельные конструкции, имеющие жесткий, сохраняющий форму каркас. Обширный класс ригидных объектов составляют, например, автомобили самых разных моделей, разыскиваемых или отслеживаемых различными структурами. Для их эффективного автоматизированного обнаружения предлагается применить разработанные в ИПС им. А.К. Айламазяна РАН инструментальные средства, объединенные под общим названием «ПС НСКиД» [7]. Программная система имеет модульную структуру и содержит все многообразие средств фильтрации и обработки изображений. Доступные модули могут быть применены для решения задач классификации и идентификации с применением высокопроизводительных вычислителей. Существенный вклад в развитие алгоритмических и программных средств системы «ПС НСКиД» внесен сотрудниками ИСА РАН, которые разработали ряд алгоритмов для распараллеливания системного и прикладного программного обеспечения.

Постановка задач





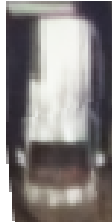
Проблема поиска сложных ригидных объектов разбивается на две взаимосвязанные задачи: поиск зон интереса и распознавание с визуализацией результатов. В первой задаче осуществляется предварительная обработка изображений ДЗЗ модулем «закраски» (выполняется поиск зон, близких к текстурам крыш заданных моделей автомобилей [4]),

далее связанные зоны пикселей подвергаются грубой фильтрации по соотношению сторон; общей площади обнаруженного фрагмента автомобиля; степени заполнения подобласти пикселями, отнесенными к классам обучающей выборки, и пр. Во второй задаче происходит непосредственное распознавание выявленных зон с помощью сверточной нейронной сети глубокого обучения и сохранение результатов поиска автомобилей.

Для проведения исследований использовались снимки размером 13236x20122 пикселя. Экспертом готовилась обучающая выборка из определенного числа изображений автомобилей (таблица 1). При этом отсутствуют сведения о других моделях автомобилей и других объектах на снимке (например, зданиях и деревьях). Такая постановка задачи наиболее близка к реальной жизненной ситуации, так как чаще всего на снимке ДЗЗ необходимо искать не все автомобили сразу, а лишь только конкретные модели.

Таблица 1

Обучающая выборка из выбранных экспертом изображений

| № эталонного изображения | 1 | 2 | 3 | 4 | 5 |
|--------------------------|--|--|--|--|--|
| Изображение |  |  |  |  |  |

Экспериментальное решение задач в системе «ПС НСКиД»

Программа в рамках «ПС НСКиД», ориентированная на конвейерно-параллельную архитектуру, может использовать различные виды параллелизма, такие как

- параллелизм независимых ветвей (в рамках одной программы могут быть выделены независимые части программы, не имеющие зависимостей, которые могут исполняться параллельно);

- параллелизм множества объектов или данных (обработка информации о различных, но однотипных объектах по одному и тому же или почти по одному и тому же алгоритму);

- параллелизм смежных операций (использование принципа конвейеризации вычислительных операций).

Модули программы имеют поддержку неграфических вычислений (в т.ч. с использованием OpenCL и CUDA).

Работа «ПС НСКиД» основывается на подсистеме диспетчеризации команд, основной задачей которой является управление процессом передачи данных между модулями и

инициация их запуска путем передачи вычислителям управляющих команд. Общий алгоритм работы системы диспетчеризации можно описать следующим образом:

- 1) построение начального состояния (подготовка множества команд инициализации вычислителей и команд запуска модулей-«поставщиков» ресурсов, загрузка модулей, необходимых для решения прикладной задачи);
- 2) построение списка применимых команд;
- 3) переход к п. (7) в случае отсутствия применимых команд;
- 4) выбор применимой команды, обладающей наивысшим приоритетом (приоритет команды прямо зависит от количества аппаратных ресурсов, требуемых для ее исполнения, выбор наиболее ресурсоемкой команды в каждом состоянии максимизирует нагрузку на узлы или ядра высокопроизводительного вычислителя, например кластерного вычислительного устройства);
- 5) передача выбранной приоритетной команды вычислителю;
- 6) обновление внутреннего состояния системы, далее переход к п. (2);
- 7) при наличии незавершенных команд, переданных вычислителям:
 - ожидание подтверждения от вычислителей;
 - обновление внутреннего состояния системы, далее переход к п. (2);
- 8) завершение работы.

Предложенный подход к диспетчеризации работы, основывающийся на «жадном» алгоритме локально-оптимального выбора применимых команд, несмотря на его простоту, является достаточно эффективным решением, учитывающим ограничения на количество используемых аппаратных ресурсов.

Формат декларативного описания схем решаемых задач должен соответствовать синтаксису описаний задач для системы «ПС НСКиД». Схемы решаемых задач содержат информацию о требуемых вычислительных модулях, их настройках и каналах передачи данных. Общая структура описания задачи представлена в таблице 2.

Таблица 2

Структура описания задачи

```
<task xmlns="http://tempuri.org/KernelTask.xsd">
  <modules>
    <module internalname="ModuleNameA" name="moduleA">
      <var name="...">...</var>
      ...
    </module>
    <module internalname="ModuleNameB" name="moduleB">
      <var name="...">...</var>
      ...
    </module>
  </modules>
  <channels>
    <channel>
```

```
<out>ModuleNameA.ChannelXName</out>
<in>ModuleNameB.ChannelYName</in>
</channel>
...
</channels>
</task>
```

Теги и атрибуты этого описания имеют следующие значения:

- <task> – корневой тег описания;
- <modules> – секция описания использованных модулей;
- <module> – описание требуемого для решения задачи модуля;
- атрибут “internalname” внутри тега <module> – уникальное (в пределах описания задачи) «внутреннее имя» модуля;
- атрибут “name” внутри тега <module> – имя библиотеки, содержащей реализацию использованного модуля;
- <var> – тег для установки значения параметра модуля, имя параметра задается атрибутом “name”;
- <channels> – секция описания каналов передачи данных;
- <channel> – добавление тега создает новый канал передачи данных, источник данных задается в подтеге <out>, место назначения – в подтеге <in> (в схеме “ChannelXName” – имя точки подключения канала для отправки данных от модуля “ModuleNameA”, а “ChannelYName” – имя точки подключения для получения данных модулем “ModuleNameB”).

Задача поиска зон интереса

Схема решения задачи представлена в таблице 3. Она включает следующие модули:

- модули чтения обрабатываемых изображений (findpng);
- GPU-версия модуля «закраски» (emarking_magma);
- модуль чтения эталонных изображений, заданных экспертом (findpng_stream);
- модуль маркировки связанных зон пикселей (mark_data_entities_from_matrix);
- модуль извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний (get_data_entities);
- модуль визуализации результатов обработки на исходных данных (scanwindowmark);
- модуль сохранения результатов обработки в виде изображений (savepng);
- модуль сохранения описаний найденных зон интереса (savexml).

Таблица 3

Описание схемы первой задачи

```
<task xmlns="http://tempuri.org/KernelTask.xsd">
  <modules>
    <module internalname="Objects" name="findpng">
      <var name="path">./tasks/snimok/scan.xml</var>
```

```

    <var name="split">2</var>
</module>
<module internalname="ObjectsNonChanged" name="findpng">
  <var name="path">./tasks/snimok/scan.xml</var>
  <var name="split">1</var>
</module>
<module internalname="Emarking" name="emarking_magma">
  <var name="path">./tasks/snimok/etalons.xml</var>
  <var name="bgclass">1 2 3 4 5</var>
  <var name="split_etalons">5</var>
  <var name="mode">3</var>
  <var name="threshold">0.0043</var>
</module>
<module internalname="Find PNG" name="findpng_stream">
  <var name="path">./tasks/snimok/etalons.xml</var>
  <var name="max_images">1</var>
</module>
<module internalname="mark_data" name="mark_data_entities_from_matrix">
  <var name="region_type">0</var>
  <var name="bg_value">0 6</var>
</module>
<module internalname="get_data" name="get_data_entities">
  <var name="delete_sizes">{0..99},{1301..1000000000}</var>
  <var name="proportion">2.5</var>
  <var name="min_alpha">0</var>
  <var name="max_alpha">0.65</var>
</module>
<module internalname="DebugMarker" name="scanwindowmark" />
<module internalname="ResultWriter" name="savepng">
  <var name="path">./tasks/snimok/output/result/</var>
</module>
<module internalname="XMLWriter" name="savexml">
  <var name="path">./tasks/snimok/output/results.xml</var>
</module>
</modules>
<channels>
<channel>
  <out>Objects.Output</out>
  <in>Emarking.Input</in>
</channel>
<channel>
  <out>Find PNG.Output</out>
  <in>Emarking.InputLearn</in>
</channel>
<channel>
  <out>Emarking.MatrixOutput</out>
  <in>mark_data.Input</in>
</channel>
<channel>
  <out>mark_data.Output</out>
  <in>get_data.InputMatrix</in>
</channel>
<channel>
  <out>Objects.Output</out>
  <in>get_data.InputImage</in>
</channel>
<channel>
  <out>ObjectsNonChanged.Output</out>
  <in>DebugMarker.InputImage</in>
</channel>
<channel>
  <out>DebugMarker.Output</out>
  <in>ResultWriter.Input</in>
</channel>
<channel>
  <out>get_data.Results</out>
  <in>DebugMarker.InputResults</in>
</channel>

```

```
<channel>  
<out>get_data.Results</out>  
<in>XMLWriter.Input</in>  
</channel>  
</channels>  
</task>
```

Цепочка обработки, выполняемая в параллельно-конвейерном режиме, содержит девять каналов передачи данных:

– от модуля чтения обрабатываемых изображений (псевдоним модуля (internalname) в рамках схемы – “Objects”) к модулю «закраски» (псевдоним “Emarking”), входной канал Input;

– от модуля чтения эталонных изображений (псевдоним “Find PNG”) к модулю «закраски», входной канал InputLearn;

– от модуля «закраски» к модулю маркировки связанных зон пикселей (псевдоним “mark_data”), входной канал Input;

– от модуля маркировки связанных зон пикселей к модулю извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний (псевдоним “get_data”), входной канал InputMatrix;

– от модуля чтения обрабатываемых изображений (псевдоним “ObjectsNonChanged”) к модулю извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний, входной канал InputImage;

– от модуля чтения обрабатываемых изображений (псевдоним “ObjectsNonChanged”) к модулю визуализации результатов обработки (псевдоним “DebugMarker”), входной канал InputImage;

– от модуля визуализации результатов обработки к модулю сохранения результатов обработки в виде изображений (псевдоним “ResultWriter”), входной канал Input;

– от модуля извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний к модулю визуализации результатов обработки, входной канал InputResults;

– от модуля извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний к модулю сохранения описаний найденных зон интереса (псевдоним “XMLWriter”).


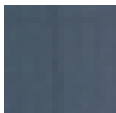

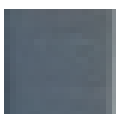
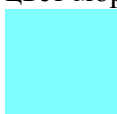

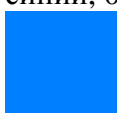

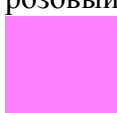

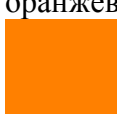
Особенности функционирования модулей:

– модуль «закраски» имеет параллельную реализацию процесса обучения (средствами пользовательского программного кода) и параллельную обработку поступающих на «закраску» изображений (динамическая балансировка нагрузки, реализованная средствами вычислительного ядра);

– модуль «закраски» разбивает эталонные изображения крыш эталонных автомобилей на пересекающиеся друг с другом зоны размером 5x5 пикселей (таблицы 1 и 4), каждая из которых содержит 5 (ширина)x5(высота)x3(число каналов в изображении) = 75 информативных признаков (размеры зоны определяются параметром “split_etalons”);

Таблица 4

Описание базы знаний для закрашки областей интереса

| № класса | Вид региона интереса | Эталонное изображение крыши автомобиля | Назначенный цвет «закраски» |
|----------|---|---|--|
| 6 | Малоинформативные области (распознаваемые с малой уверенностью) | – | черный, RGB: 0 0 0  |
| 1 | Крыша автомобиля 1 |  | салатовый, 0 255 128  |
| 2 | Крыша автомобиля 2 |  | цвет морской волны, 128 255 255  |
| 3 | Крыша автомобиля 3 |  | синий, 0 128 255  |
| 4 | Крыша автомобиля 4 |  | розовый, 255 128 255  |
| 5 | Крыша автомобиля 5 |  | оранжевый, 255 128 0  |

– параметр “threshold” модуля «закраски» позволяет задать минимальное относительное расстояние до класса региона, в случае если классификатор не дает необходимой уверенности, пиксель результирующего изображения закрашивается в черный цвет (класс 6);

– для лучшего представления результатов обработки и для сокращения одномоментного использования оперативной памяти исходный снимок был разбит на четыре фрагмента (параметр “split”, определяющий количество разбиений снимка по горизонтали и вертикали), каждый фрагмент далее обрабатывается как независимое изображение, при этом информация о том, откуда взят отдельный фрагмент, сохраняется до конца обработки;

– модуль «закраски» работает с пятью классами регионов интереса (используются выделенные экспертом крыши автомобилей), примеры результатов закраски (отдельные фрагменты) приведены в таблице 5;

Таблица 5

Примеры «закраски»

| Фрагмент изображения | исходного | Фрагмент результирующего изображения |
|---|-----------|--|
|  | |  |
|  | |  |
|  | |  |

– модуль чтения исходных изображений в немодифицированном (неразделенном) виде необходим для того, чтобы потом на эти изображения нанести результаты обработки;

– модуль «закраски» функционирует в режиме выделения целевых регионов в виде нанесенных на исходное изображение меток, дополнительно таблица с метками классов передается модулю маркировки связанных зон пикселей (mark_data);

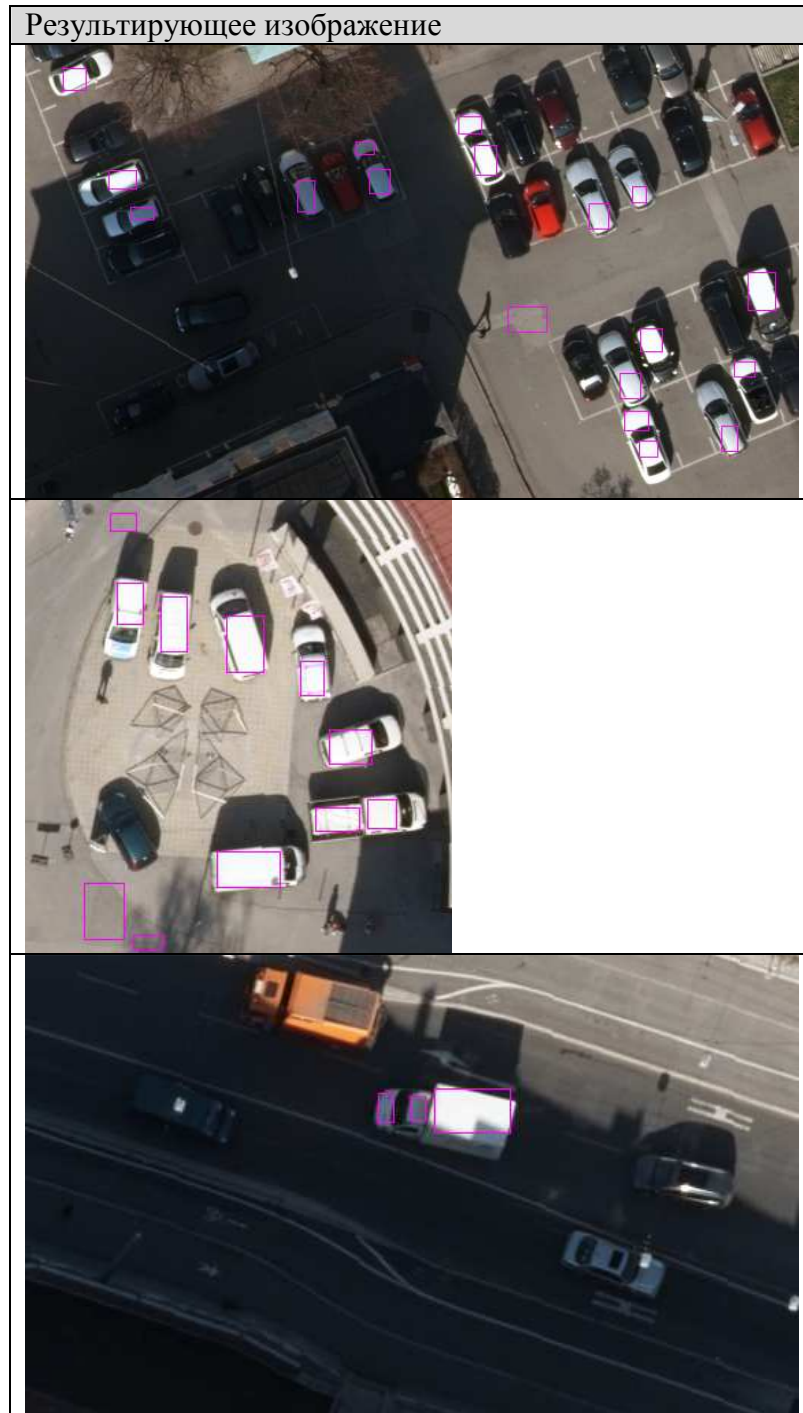
– модуль извлечения зон интереса в виде изображений, матриц идентификаторов классов и/или описаний (get_data_entities) осуществляет извлечение зон интереса в виде изображений и их описаний и осуществляет грубую фильтрацию обнаруженных объектов;

– модуль визуализации результатов обработки (DebugMarker) рисует на копии исходного изображения фиолетовые рамки, на данном этапе это рамки вокруг найденных зон интереса,

удовлетворяющих параметрам грубой фильтрации; результаты визуализации приведены в таблице 6, на них могут присутствовать ошибочно выделенные области, удалению которых посвящена вторая задача;

Таблица 6

Примеры выделенных зон интереса



– пример файла, получаемого от модуля сохранения результатов поиска зон интереса, приведен в таблице 7.

Таблица 7

Фрагмент файла с информацией о найденных зонах интереса

```

<RecognitionResults xmlns="http://tempuri.org/RecognitionResults.xsd">
  <object>
    <path>./tasks/snimok/input_images/0072_modified.png</path>
    <class>0</class>
    <xpos>3339</xpos>
    <ypos>28</ypos>
    <width>27</width>
    <height>55</height>
    <prec>0.00</prec>
    <id>0</id>
  </object>
  ...
</RecognitionResults>

```

Согласно схеме, список директорий с обрабатываемыми изображениями приведен в файле “./tasks/snimok/scan.xml” (параметр “path” модулей “Objects” и “ObjectsNonChanged”, таблица 8). Рекурсивное использование поддиректорий отключено.

Таблица 8

Список директорий с обрабатываемыми изображениями

```

<dirs xmlns="http://tempuri.org/Input.xsd">
  <dir path = "./tasks/snimok/input_images/" />
  ...
</dirs>

```

Список эталонных изображений крыш автомобилей, подготовленных экспертом, оформляется схожим образом, однако там дополнительно присутствуют идентификаторы классов и указания на то, каким цветом на результирующем изображении закрашивать пиксели, отнесенные к соответствующему классу (параметр “path” модуля “Emarking”, таблица 9, файл “./tasks/snimok/etalons.xml”).

Таблица 9

Список директорий с обрабатываемыми изображениями

```

<dirs xmlns="http://tempuri.org/Input.xsd">
  <dir classid = "1" r_color = "0" g_color = "255" b_color = "128"
    path = "./tasks/snimok/input_images/classes/1/" />
  <dir classid = "2" r_color = "128" g_color = "255" b_color = "255"
    path = "./tasks/snimok/input_images/classes/2/" />
  <dir classid = "3" r_color = "0" g_color = "128" b_color = "255"
    path = "./tasks/snimok/input_images/classes/3/" />
  <dir classid = "4" r_color = "255" g_color = "128" b_color = "255"
    path = "./tasks/snimok/input_images/classes/4/" />
  <dir classid = "5" r_color = "255" g_color = "128" b_color = "0"
    path = "./tasks/snimok/input_images/classes/5/" />
</dirs>

```

Следует отметить метод распараллеливания программ, предложенный в ИСА РАН [2]. В рамках подхода используется принцип расширения модели многогранников для устройств с динамическим параллелизмом, позволяющий осуществлять распараллеливание программ для универсальных многоядерных и графических процессоров. В алгоритме «закраски» для графического процессора выполняются биекция координат пикселей в индексы cuda-поток. При этом реализуется когерентный доступ к глобальной памяти, что позволяет избежать нежелательных транзакций и эффективно использовать ресурсы ускорителя вычислений.

Код классификатора генерируется автоматически с использованием макросов, производящих подсчет расстояния Евклида-Махаланобиса [8] путем явного раскрытия матрично-векторных операций. Значения обратной модифицированной матрицы ковариаций подставляются как константы в арифметические операции, составляющие вычисление раскрытых матрично-векторных произведений. Такой подход выгоден тем, что вообще не требуется выделять память под значения эталонных векторов и обратных матриц, что при повышенном количестве инструкций, оперирующих с константами, и увеличении нагрузки на кэш инструкций позволяет интенсивно использовать регистры и быстрые multiply-add-инструкции, оперирующие над уже готовыми данными в регистрах [1; 6].

В подходе, предложенном в ИПС им. А.К. Айламазяна РАН, для эффективного решения задачи «закраски» обрабатываемое изображение представляется в виде grid-сети, каждый элемент этой сети – блок (в терминологии CUDA) размером $s \times s$. Размеры блока, определяющие число вычислительных cuda-потоков (по s^2 потоков), выбираются автоматически, исходя из размеров используемого для «закраски» сканирующего окна и аппаратных характеристик задействованного ускорителя вычислений. Обработка реализована таким образом, что каждый поток работает со своим отдельным пикселем исходного изображения (и, соответственно, с ассоциируемой с ним окрестностью). Для ускорения счета в быструю shared-память графического ускорителя загружаются только те данные, что необходимы для работы отдельного блока, т.е. достаточные для обработки s^2 пикселей, зарезервированных за ним (учитываются выступающие границы сканирующих окон, одновременно обрабатывается сразу несколько таких блоков). Остальные данные хранятся в обычной памяти графического ускорителя [3; 5].

Задача распознавания с визуализацией результатов

Схема решения задачи представлена в таблице 10. Она включает следующие модули:

- модуль чтения сохраненных зон интереса (read_recresults);
- модуль GPU-версии сверточной нейронной сети (nnforge_cnn_cuda), реализован с использованием библиотеки nnForge [9];
- модуль чтения обрабатываемых изображений (findpng)
- модуль визуализации результатов обработки на исходных данных (scanwindowmark);
- модуль сохранения результатов обработки в виде изображений (savepng);
- модуль сохранения описаний найденных зон интереса (savexml).

Таблица 10

Описание схемы второй задачи

```
<task xmlns="http://tempuri.org/KernelTask.xsd">
  <modules>
    <module internalname="Objects" name="read_recresults">
```

```

<var name="path">./tasks/snimok/output/results.xml</var>
<var name="stretch">3.0</var>
<var name="max_images" type="int">1000</var>
</module>
<module internalname="Net" name="nnforge_cnn_cuda">
  <var name="dir">./tasks/snimok_res/</var>
  <var name="number_of_classes">5</var>
  <var name="max_rotation_angle_in_degrees" type="float">40.0</var>
  <var name="max_scale_factor" type="float">1.5</var>
  <var name="max_shift" type="float">10.0</var>
  <var name="max_contrast_factor" type="float">1.25</var>
  <var name="max_brightness_shift" type="float">30.0</var>
  <var name="random_sample_count" type="int">750</var>
  <var name="growing_scaling" type="int">2</var>
  <var name="limit" type="float">0.95</var>
  <var name="schema_type" type="int">0</var>
  <var name="mem_multiplier" type="float">4.0</var>
</module>
<module internalname="MAP" name="findpng">
  <var name="path">./tasks/snimok/scan.xml</var>
  <var name="split">1</var>
</module>
<module internalname="DebugMarker" name="scanwindowmark" />
<module internalname="ResultWriter" name="savepng">
  <var name="path">./tasks/snimok_res/output/</var>
</module>
<module internalname="XMLWriter" name="savexml">
  <var name="path">./tasks/snimok_res/output/results.xml</var>
</module>
</modules>
<channels>
<channel>
  <out>Objects.OutputArray</out>
  <in>Net.InputRecognizeArray</in>
</channel>
<channel>
  <out>Net.Output</out>
  <in>DebugMarker.InputResults</in>
</channel>
<channel>
  <out>Net.Output</out>
  <in>XMLWriter.Input</in>
</channel>
<channel>
  <out>MAP.Output</out>
  <in>DebugMarker.InputImage</in>
</channel>
<channel>
  <out>DebugMarker.Output</out>
  <in>ResultWriter.Input</in>
</channel>
</channels>
</task>

```

Цепочка обработки, выполняемая в параллельно-конвейерном режиме, содержит пять каналов передачи данных:

- от модуля чтения сохраненных зон интереса (псевдоним “Objects”) к модулю искусственной нейронной сети (псевдоним “Net”), входной канал InputRecognizeArray;
- от модуля искусственной нейронной сети к модулю визуализации результатов обработки на исходных данных (псевдоним “DebugMarker”), входной канал InputResults;
- от модуля искусственной нейронной сети к модулю сохранения описаний найденных зон интереса (псевдоним “XMLWriter”), входной канал Input;

– от модуля чтения обрабатываемых изображений (псевдоним “Map”) к модулю визуализации результатов обработки на исходных данных, входной канал InputImage;

– от модуля визуализации результатов обработки на исходных данных к модулю сохранения описаний найденных зон интереса (псевдоним “ResultWriter”).

Особенности функционирования модулей:

– модуль чтения сохраненных зон интереса (“Objects”) считывает и отправляет на обработку фрагменты исходных изображений, описанные в файле “./tasks/snimok/output/results.xml”, полученном при решении первой задачи;

– модуль чтения сохраненных зон интереса (“Objects”) захватывает дополнительные области исходного изображения, например если значение параметра “stretch” равно “3.0”, то зона увеличивается в три раза по высоте и ширине, при этом исходная область оказывается в центре, это делается для того, чтобы в обрабатываемую область попала не только выделенная крыша автомобиля (или ее фрагмент), но и все транспортное средство целиком (как проиллюстрировано на рис. 1);

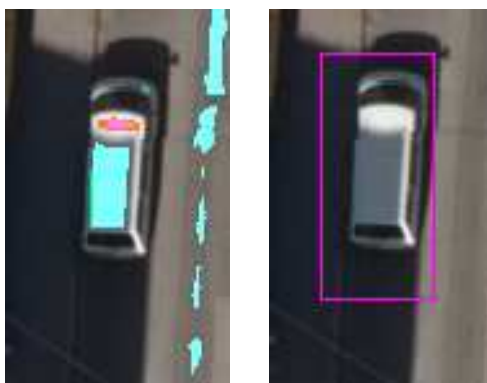


Рис. 1. Иллюстрация алгоритма выделения регионов для дальнейшей обработки (расширенная зона интереса).

– модуль чтения сохраненных зон интереса (“Objects”) для сокращения пересылок данных работает в режиме буферизации, количество передаваемых за одну отправку изображений задается параметром “max_images”;

– метод распознавания заключается в построении нейронной сети глубокого обучения, в которой применяются слои разных типов: сверточные и субдискретизирующие (в исследовании использовалось восемь слоев, однако можно выбрать любое их число); также применен слой извлечения контраста; сверточные слои отвечают за выделение особенностей в выходах предыдущего слоя (например, выделяют вертикальные линии и углы); субдискретизирующие слои предназначены для уменьшения размерности вектора признаков различными алгоритмами (вычисление скользящего среднего, локального максимума и т.п.);

– модуль визуализации результатов обработки (DebugMarker) рисует на копии исходного изображения фиолетовые рамки вокруг найденных зон интереса, распознанных искусственной нейронной сетью с высокой уверенностью.

В результате обработки выделенных зон с помощью сверточной нейронной сети, на выходе получаем координаты лишь целевых ригидных объектов, соответствующих обучающей выборке, каждый объект выделяется на исходном изображении фиолетовой рамкой. Разработанные схемы обработки могут быть использованы практически без модификаций и в других задачах обработки данных ДЗЗ. Конвейерно-параллельный механизм организации вычислений и буферизованная отправка пакетов данных позволяют эффективно загрузить все узлы используемого вычислительного устройства.

Графический интерфейс программно-инструментальной системы

В результате совместных исследований был разработан специальный графический интерфейс для эффективного использования компонентов программно-инструментальной системы, позволяющий осуществлять выполнение обработки изображений ДЗЗ, в том числе

- автоматическую кластеризацию;
- поиск и классификацию объектов на снимках ДЗЗ (с помощью статистических и нейросетевых классификаторов – на основе сверточной нейронной сети, нейронной сети прямого распространения, Хопфилда, Хемминга, Кохонена и вероятностной и др.);
- спектрографическую «закраску» полноцветных и мультиспектральных изображений;
- улучшение качества изображений за счет их фильтрации;
- выделение линий и контуров обнаруженных объектов;
- сжатие изображений.

Интерфейс реализует поддержку функций универсальной моделирующей среды, обеспечивает задание необходимого набора вычислительных модулей и выбор их настроек; конфигурирование каналов связи; запуск и остановку распределенных вычислений; сохранение и визуализацию результатов. При этом последовательность действий, приводящая к желаемому результату, минимальна. Обеспечивается максимальная гибкость формирования решаемых задач.

Графический интерфейс позволяет выбрать ранее созданные схемы заданий и создавать новые; отслеживать корректность создаваемых схем (соответствие синтаксиса декларативного описания); переключаться между предыдущими вариантами описания задачи; осуществлять запуск новых вычислений и остановку ранее запущенных.

Индикатор статуса корректности схемы обладает собственной панелью сообщений, расположенной в нижней части интерфейса. В ней появляются сообщения, предупреждающие пользователя о появившихся ошибках, сообщения о корректности

входных данных и др. сообщения, связанные с обработкой текста схемы разрабатываемой задачи. Средняя часть интерфейса представлена двумя формами: “Read and Save modules” и “Task definition (XML)”. Первая форма представлена на рис. 2, вторая – на рис. 3-5.

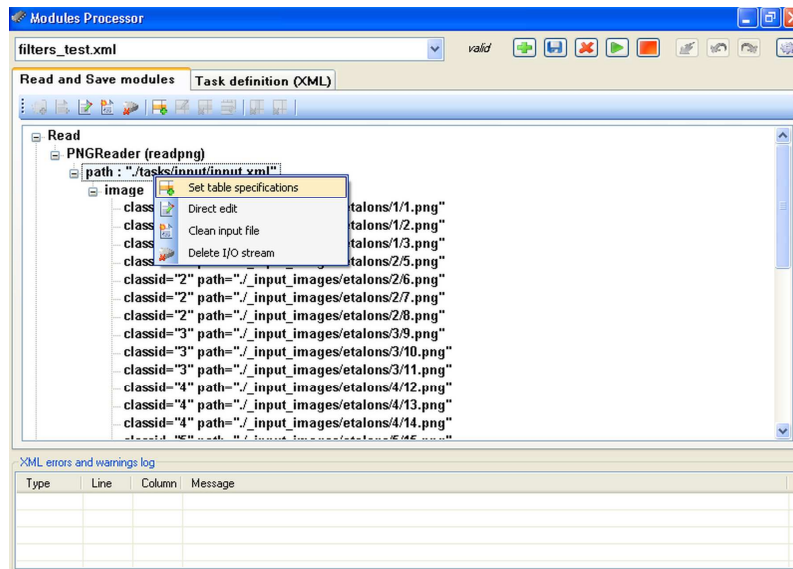


Рис. 2. Форма работы с входными и выходными данными.

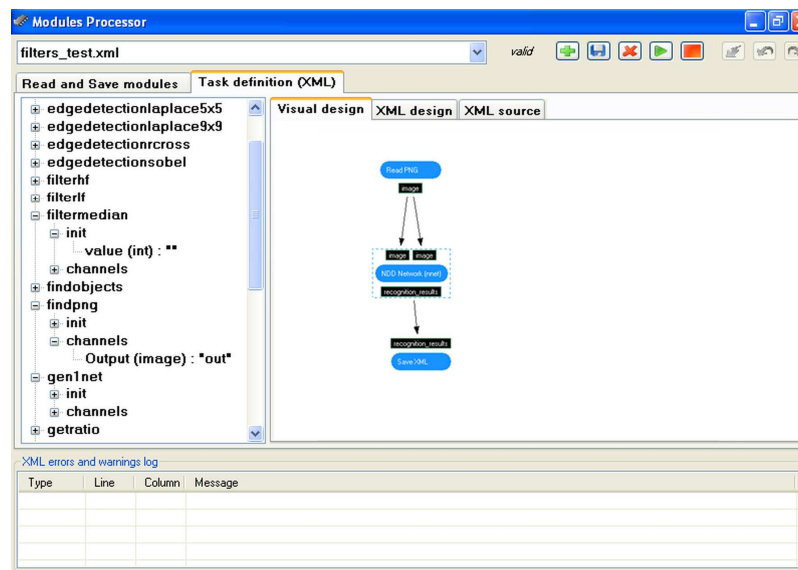


Рис. 3. Форма представления задания в виде функциональных блоков.

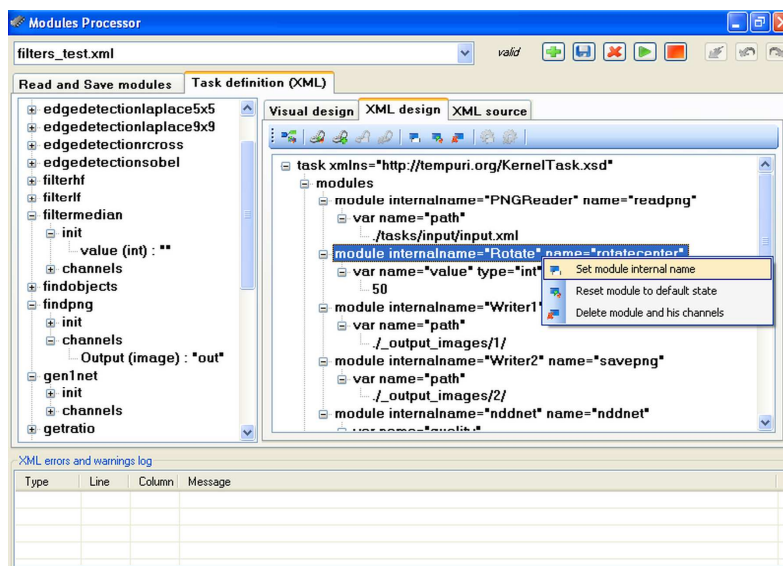


Рис. 4. Форма представления задания в виде дерева.

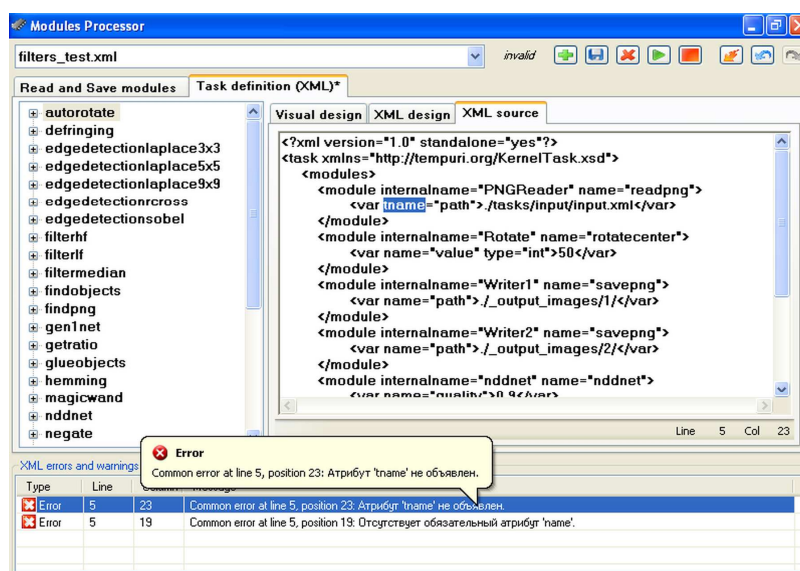


Рис. 5. Форма представления задания в текстовом виде.

Форма работы с входными и выходными данными “Read and Save modules” используется для выбора и изменения обрабатываемых файлов (например, если они представлены в виде мультиспектральных изображений или в форматах XML, TXT, DAT) и задания указаний модулям сохранения результатов. На вкладке отображаются настройки тех модулей, что имеют только входные (подключены к ветви “Read” дерева данных) или выходные каналы (подключены к ветви “Save”).

Используя тулбар и контекстное меню, пользователь получает набор функций, позволяющий манипулировать входными и выходными данными задания, не влияя на его структуру:

- создание и удаление файлов или директорий;
- создание файлов XML, TXT и DAT с данными;

- прямая правка существующих файлов с данными с помощью встроенного редактора;
- добавление, правка и удаление записей в XML-файлах;
- перемещение табличных записей внутри XML-файлов;
- работа с атрибутами записей.

Форма работы с текстом задачи “Task definition (XML)” позволяет создавать новые схемы заданий и редактировать существующие. Она содержит две области. В первой – дерево доступных модулей, во второй – область представления задачи.

Дерево доступных модулей представляет собой список доступных для применения модулей, для каждого модуля приводится информация о наборе его параметров и их значениях по-умолчанию, информация о реализованных каналах передачи данных. Заполнение этого дерева происходит на основе сканирования рабочих директорий системы на наличие файлов, описывающих реализованные модули.

Область представления задачи содержит доступные пользователю инструменты для управления схемой задачи. Пользователю предоставляется возможность реализации задуманного им алгоритма с помощью графической визуализации на основе функциональных блоков, дерева задания или текстового редактора. Разработанные инструменты имеют обратную связь: изменение в рабочем пространстве одного инструмента влияет на рабочие пространства других инструментов.

Форма представления задания в виде функциональных блоков (рис. 3) предназначена для визуального редактирования схемы задачи. Функциональные блоки перетаскиваются на рабочую область с помощью манипулятора типа «мышь». Аналогичным образом добавляются каналы связи между модулями. Модули отображаются в виде геометрических фигур, например в виде овала, прямоугольника, равнобедренной трапеции. Входные и выходные каналы визуализируются в виде черных прямоугольников.

Форма представления задания в виде дерева (рис. 4) позволяет производить редактирование схемы задачи путем добавления и удаления поддеревьев в основном дереве задачи с помощью drag-and-drop-операций манипулятором типа «мышь». Дерево делится на две области: область модулей и область каналов. В области модулей пользователем создается набор необходимых для решения поставленной задачи модулей. Там же можно настраивать все доступные параметры, замещая их умолчательные значения. Редактор обладает интеллектуальностью: добавить поддерево можно лишь в разрешенную область главного дерева, при инициализации drag-and-drop-события включается подсветка цветом тех мест, куда можно добавить интересующие пользователя данные. В области каналов устанавливаются связи между ранее добавленными модулями. Настройка каналов производится на основе информации о доступных каналах связываемых модулей.

Используя тулбар и контекстное меню дерева задания, пользователь получает набор функций редактирования текста задачи:

- сброс текста задачи в начальное состояние (без модулей и связей между ними);
- создание, правка и удаление каналов связи между модулями;
- установка нового внутреннего имени для модуля;
- сброс настроек модуля в умолчательное состояние (как для отдельных параметров, так и всех сразу);
- удаление модуля вместе с объявленными каналами.

Форма представления задания в текстовом виде (рис. 5) позволяет производить редактирование схемы задачи как с помощью drag-and-drop-операций манипулятором типа «мышь» (например, «перетаскивая» в текст задачи модуль из дерева доступных модулей), так и прямым способом, то есть путем набора текста на клавиатуре. Такой подход потенциально более сложен, поэтому в помощь пользователю приходит панель сообщений, сигнализирующая о допущенных ошибках.

Заключение

Настоящая работа подводит некоторый итог совместных предварительных исследований. Получено решение проблемы поиска сложных ригидных объектов. В том числе разработаны следующие алгоритмы: поиск зон, близких к текстурам крыш заданных моделей автомобилей; промежуточная фильтрация по соотношению сторон, общей площади обнаруженного фрагмента и пр.; распознавание с помощью сверточной нейронной сети глубокого обучения. Конвейерно-параллельное исполнение реализаций алгоритмов обладает близким к линейному масштабированием.

Разработанные алгоритмы «закраски» могут быть применены при решении схожих задач. Представленный графический интерфейс позволяет на ходу добавлять в схему задачи новые блоки, реализующие дополнительный функционал. Дальнейшие исследования предполагают развитие этого графического интерфейса, добавление многопользовательского режима функционирования вычислительного ядра и разработку новых высокоэффективных алгоритмов обработки данных дистанционного зондирования.

Список литературы

1. Лебедев А.С. Классификация мультиспектральных снимков дистанционного зондирования Земли с использованием метрики Евклида-Махаланобиса // Научно-информационные технологии : труды XVII Молодежной научно-практической конференции

SIT-2013 / УГП имени А.К. Айламазяна. — Переславль-Залесский : Университет города Переславля, 2013. — С. 123-130.

2. Лебедев А.С. Оптимизация временной локальности данных при автоматическом распараллеливании линейных программ // Современные проблемы науки и образования. — 2014. — № 6 [Электронный ресурс]. — URL: <http://www.science-education.ru/120-16255> (дата обращения: 23.03.2015).

3. Попков А.Ю., Соченков И.В., Хачумов В.М. Гетерогенный вычислительный комплекс для решения инженерных и научных задач // Сборник тезисов докладов Третьего национального суперкомпьютерного форума (НСКФ-2014) (Переславль-Залесский, 25-27 ноября 2014 г.). — Переславль-Залесский : Институт программных систем им. А.К. Айламазяна Российской академии наук, 2014 [Электронный ресурс]. — URL: <https://docs.google.com/uc?export=download&id=0B-Qay3kEFxqfZTJLeEgybVnKOWc> (дата обращения: 23.03.2015).

4. Фраленко В.П. Методы текстурного анализа изображений, обработка данных дистанционного зондирования Земли // Программные системы: теория и приложения : электрон. научн. журн. — 2014. — № 4 (22). — С. 19-39.

5. Фраленко В.П. Спектрографическая «закраска» полноцветных и мультиспектральных изображений (ПС "Emarking") : Свидетельство о государственной регистрации программы для ЭВМ № 2014611689, дата приоритета: 11 декабря 2013 года.

6. Хачумов В.М., Лебедев А.С. Высокопроизводительная обработка изображений // Сборник трудов конференции «Перспективные разработки науки и техники — 2014» (Пшемысль, Польша, 7-15 ноября 2014 г.). — С. 13-16 [Электронный ресурс]. — URL: <https://docs.google.com/uc?export=download&id=0B-Qay3kEFxqfZGdIZUpWcXM2UU0> (дата обращения: 23.03.2015).

7. Хачумов В.М., Тищенко И.П., Талалаев А.А., Константинов К.А., Фраленко В.П., Емельянова Ю.Г. Нейросетевая система контроля телеметрической информации, диагностики подсистем космических аппаратов, обработки космических снимков (ПС НСКид) : Свидетельство о государственной регистрации программы для ЭВМ № 2012613261, дата приоритета: 18.11.2011, дата регистрации: 06.04.2012.

8. Khachumov M.V. Distances, Metrics and Cluster Analysis // Scientific and Technical Information Processing. — 2012. — Vol. 39, № 6. — P. 1-7.

9. nnForge — Convolutional and fully-connected neural networks C++ library [Электронный ресурс]. — URL: <http://milakov.github.io/nnForge/> (дата обращения: 23.03.2015).

Рецензенты:

Цирлин А.М., д.т.н., профессор, главный научный сотрудник Исследовательского центра системного анализа ФГБУН «Институт программных систем им. А.К. Айламазяна» Российской академии наук, с. Веськово;

Остроух А.В., д.т.н., профессор кафедры «Автоматизированные системы управления» ФГБОУ ВПО «Московский автомобильно-дорожный государственный технический университет (МАДИ)», г. Москва.