САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

АНАЛИЗ КОРРЕЛЯЦИОННОГО МЕТОДА В СОВРЕМЕННЫХ АРХИТЕКТУРАХ В ЗАДАЧЕ ВЫДЕЛЕНИЯ ОБЪЕКТОВ НА АЭРОФОТОСНИМКЕ

Иванов Александр Аркадьевич

Дипломная работа

Научный руководитель	…..... /подпись/	д.ф.-м.н., проф. Граничин О.Н.
Рецензент	…..... /подпись/	к.ф.-м.н., доц. Сысоев С.С.
"Допустить к защите" заведующий кафедрой,	…..... /подпись/	д.ф.-м.н., проф. Терехов А.Н.

Санкт-Петербург

2014

SAINT PETERSBURG STATE UNIVERSITY

Mathematics & Mechanics Faculty

Software Engineering Chair

ANALYSIS OF THE CORRELATION METHOD IN MODERN ARCHITECTURES IN THE PROBLEM OF DETECTING OBJECTS IN AN AERIAL PHOTO

by

Aleksandr Ivanov

Specialist Thesis

Supervisor	….....	Professor O.N. Granichin
Reviewer	….....	Docent S.S. Sysoev
"Approved by" Head of Department	….....	Professor A.N. Terekhov

Saint Petersburg

2014

Оглавление

Введение

В настоящее время актуальной задачей является сбор и регистрация визуальной информации местности с помощью беспилотных летательных аппаратов (далее БПЛА). Широкое применение в этой области получили сверхлегкие планеры, оснащенные системой автопилота, которая позволяет задавать траекторию полета аппарата перед его запуском при помощи специальных программных средств. Примером такой системы управления может служить бесплатная и свободная для коммерческого использования утилита *APM Mission Planner*[1], разработанная Michael Oborne под среду Microsoft .NET. Среди основных возможностей можно перечислить:

- Наличие визуального редактора маршрута, с использованием карт Google
- Настройка параметров автопилота
- Загрузка логов полета и их анализ

В связи со стремительным развитием технологий стало возможным оборудовать такие планеры наряду с базовой навигационной аппаратурой, такой как спутниковым навигационным приёмником (GPS или ГЛОНАСС), также различной периферией, включающей фотокамеру и GSM модем, которая обслуживается дополнительным программно-аппаратным модулем. Примером такого устройства может служить микрокомпьютер Gumstix[2] Overo с установленной на нем OS Linux. Подобная конфигурация значительно расширяет функциональные возможности БПЛА, позволяет устанавливать двусторонний канал беспроводной передачи данных с компьютером диспетчера и осуществлять сбор и передачу фотоматериала.

1 Mission Planner, http://planner.ardupilot.com/
2 Gumstix, Inc., https://www.gumstix.com/

Современные достижения в области компьютерного зрения и распознавания образов наряду с естественным увеличением производительных мощностей аппаратных платформ позволяют осуществлять интеллектуальную обработку непрерывных видеопотоков, "на лету" делать анализ фотоснимков наблюдаемых территорий. Задачи автоматического поиска и обнаружения востребованы в самых разных областях. Так газовые и нефтяные корпорации могут существенно сократить свои расходы на повседневное обеспечение безопасности своих трубопроводов, благодаря возможности частичного замещения человеческого труда роботами, которые в состоянии преодолевать большие расстояния и собирать информацию о возможных повреждениях на линиях. Структуры по надзору и охране заповедных зон могут своевременно выявлять случаи пожаров, вырубки лесов. Также за последние десятилетия участились случаи крушения гражданских авиалайнеров и пропажи туристов в труднодоступных зонах.

В общем виде задача статического обнаружения некоторого объекта обуславливается выбором изображения этого объекта, в дальнейшем именуемого *шаблоном*, и фотографии местности, в которой осуществляется поиск. До недавнего времени в силу высокой затратности проведения вычислений базовой реализации алгоритма поиска точки максимальной корреляции шаблона на изображении, настоящий метод не рассматривался для практического применения в качестве эффективного способа обнаружения. В английской литературе этот подход получил название *"Template Matching"*, что буквально означает *"сопоставление шаблона"*.

Объём публикаций на эту тему значительно уступает по количеству другим технологиям в сфере распознавания образов. Так за последние десятилетия были получены весомые результаты в обучении классификаторов,

направленных на решение задач детектирования сложных объектов на фотографиях. По части распознавания лиц в работе [1] приводится исчерпывающее описание ранних исследований в этой области. Краткий курс [2] предоставляет обзор более современных подходов, среди которых стоит отметить *метод построения опорных векторов (SVM)* [3], который переводит исходные вектора в пространство более высокой размерности и ищет разделяющую гиперплоскость с максимальным зазором в этом пространстве (в случае обнаружения лиц, вектора в одной половине гиперплоскости соответственно характеризуют признаки лица, в то время как в другой все иные объекты). Текущий метод использовался исследователями как для обнаружения лиц, так и для их распознавания. Так же были получены существенные результаты по выделению на изображениях пешеходов, машин и иных объектов составной природы. Из известных работ можно отметить [4], где для разделения на гиперплоскости SVM используются множества перекрывающихся дескрипторов *гистограммы ориентированных градиентов (HOG)*.

Однако, с недавним ростом производительных мощностей и появлением недорогих платформ для проведения параллельных вычислений стала актуальной задача исследовать более детально метод поиска максимальной корреляции (далее *корреляционный метод*). Сегодня у рядового пользователя персонального компьютера, обладающего графическим процессором от фирмы NVidia[3] существует возможность в домашних условиях эффективно распараллеливать многие вычислительные задачи, что позволяет для некоторых алгоритмов ускорять их работу в десятки раз.

Также бурное развитие получают технологии *программируемых логических интегральных схем (ПЛИС)*. Среди самых поздних исследований

3 NVidia CUDA, http://www.nvidia.com/object/cuda_home_new.html

применимости корреляционного метода можно отдельно выделить [5], где авторы приводят способ автоматической генерации кода VHDL, который решает задачу поиска методом максимальной корреляции, устойчивого к вращению, масштабированию и сдвигу, на архитектуре FPGA (*Field-Programmable Gate Array*).

Так же существенным фактором, указывающим на необходимость проведения анализа корреляционного метода, является относительно малый размер шаблонов на аэрофотоснимках. Съемка под вертикальным углом с высоты сотни метров значительно усложняет задачу поиска, поскольку в проекции геометрическая структура объектов заметно искажается, и как следствие, ведет к неприменимости широкого спектра технологий по распознаванию образов, ввиду отсутствия у шаблонов необходимого перечня характерных признаков.

Таким образом, с целью решения задачи выделения объектов на аэрофотоснимках в настоящей работе проводится анализ корреляционного метода, на основе чего будет принято решение по осуществлению эффективной реализации этого подхода в архитектуре параллельного выполнения CUDA.

1. Постановка задачи

Целью настоящей работы является создание программных средств, выполняемых на CUDA, способных эффективно решать задачу поиска объектов на аэрофотоснимках. Для достижения этой цели были выделены следующие основные подзадачи:

1. Изучить существующие подходы в области анализа изображений, сравнить и оценить их применимость к задаче выделения объектов на аэрофотоснимках.
2. Выбрать один из существующих подходов или предложить собственное решение, основанное на анализе положительных сторон текущих методов.
3. Выполнить реализацию на CUDA.
4. Апробировать реализованный подход на конкретных снимках, сделанных в ходе съёмки БПЛА, и провести сравнительный анализ по полученным данным.

2. Обзор существующих подходов

Существует множество методов, применяемых для автоматического анализа изображений. Каждый отдельно взятый подход оказывается полезным в некотором ограниченном классе задач. Однако не существует универсальной техники, призванной решать весь широкий спектр задач по распознаванию, если сравнивать со способностью человеческого мозга к анализу визуальной информации. В этом смысле можно условно разделить существующие подходы на следующие категории:

- Основанные на объектных моделях *CAD (Computer-aided Detection)*. Сюда следует отнести методы, разбивающие изображение на различимые структурные объекты и секции.
- Основанные на анализе внешних очертаний и контуров объектов.
- Основанные на выделении и сопоставлении характерных признаков искомого объекта с областью изображения, обладающей наличием этих признаков.
- Основанные на поиске максимальной корреляции интенсивностей точек шаблона и области изображения.

2.1. Методы сегментации и выделения контуров

Наряду с большинством методов распознавания для текущего класса методов также свойственно проводить начальную обработку изображения, которая включает в себя устранение шумов и видимых артефактов. Обычно для этого сглаживают картинку, используя различные фильтры со свойственной им матрицей свёртки, такие как медианный фильтр для удаления импульсного шума, либо сглаживание с использованием матрицы свёртки Гаусса. Далее

проводится сегментация изображения, то есть разбиение точек по группам, в которых присутствует некая визуальная связь между элементами. Под более широким названием процесс известен как *кластеризация.* В анализе изображений это, пожалуй, одна из наиболее ранних и хорошо изученных задач. Одна из первых работ была опубликована еще в 1970 году [6]. Эта область смежна задаче анализа изображения по разбиению его на контуры. Классическим подходом для выделении краев является алгоритм Canny [7]. Также возможно структурное сравнение по выделенным очертаниям сегмента. Для этого анализируется форма, размер, компактность и местоположение. Но чаще всего, например, в медицине при определении проблемной зоны среди органов пациента, дальнейший анализ проходит с использованием классификаторов, параметры которых настроены по большим заранее приготовленным базам изображений, которые выступают в роли обучающих множеств.

При облете территории частота и качество производимых снимков напрямую зависит от параметров установленной на борту БПЛА камеры. Но как правило, объем коллекции фотоснимков, сделанных на определенном участке ландшафта, невелик, что в свою очередь делает затруднительным использование машинного обучения с целью обнаружения интересующих объектов.

Согласно [12], оперирование информацией о форме объекта как основного критерия при сравнении двух изображений убирает из рассмотрения богатые данные об интенсивности пикселей. Подобное упрощение делает анализ чувствительным к шуму, что часто приводит к ошибочному распознаванию. Более того, такие методы неприменимы в задаче обнаружения гладких изображений.

2.2. Методы составления дескрипторов по характерным признакам

Настоящий класс методов применяется для поиска совпадения выделяемых признаков объекта и изображения. В качестве таких характерных особенностей может быть устойчивая информация об углах объекта, его краях и иных пространственных свойствах. На сегодня одними из наиболее применяемых на практике дескрипторов, являются GLOH (Gradient Location and Orientation Histogram) [8], SIFT (Scale Invariant Feature Transform) [9], а также его улучшения SURF (Speeded Up Robust Features) [10] и ASIFT (Affine-SIFT) [11]. Последний, в свою очередь, имеет свойство инвариантности к аффинным преобразованиям и наиболее интересен. Ключевые точки определяются независимо друг от друга в каждом изображении, и выработанные дескрипторы представляют каждую из таких точек. При условии достаточного количества характерных точек можно вычислить глобальное аффинное преобразование между изображениями. Эта операция возможна в предположении, что соответствующие друг другу ключевые точки на обоих изображениях могут быть обнаружены независимо, и сами дескрипторы инвариантны к аффинным преобразованиям, что допускает их сравнение.

Но, несмотря на интенсивные исследования в этой области, процедура может не сработать, в особенности, когда отсутствует необходимый для сравнения набор различимых особенностей в шаблоне или изображении. В случае аэрофотоснимков, сделанных даже с небольшой высоты в 100 метров камерой с матрицей в несколько мегапикселей, размер искомого объекта может не превышать 100 пикселей по каждому измерению. Шаблон принимает в вертикальной проекции практически плоскую форму и не обладает достаточными свойствами для успешного построения дескрипторов.

2.3. Корреляционный метод

Текущий подход имеет разные наименования в литературе. В зарубежных источниках наиболее часто встречаются названия *"image matching"* и *"template matching"*. Также иногда встречаются названия *"линейный пространственный фильтр"* или менее формально *"метод скользящего окна"*. В базовой реализации этого метода сначала выбирается некоторый участок изображения, именуемый *"шаблоном",* который представляет собой объект поиска. Обозначим через $T(x_t, y_t)$ матрицу шаблона, где (x_t, y_t) представляют координаты пикселей в нем. Тогда, если $S(x, y)$ матрица изображения, в котором осуществляется поиск шаблона, алгоритм будет считать сумму произведений интенсивностей пикселей T в S. Двигая шаблон по всему изображению, вычисляется матрица корреляций. В этом случае её элементы будут равны *SAD (Sum of Absolute Differences)*, как суммы абсолютных разностей, вычисляемых по следующей формуле:

$$SAD(x, y) = \sum_{i=0}^{T_{rows}} \sum_{j=0}^{T_{cols}} \mathrm{Diff}(x+i, y+i, i, j)$$, где

Diff модуль разности интенсивностей пикселей $S(x+i, y+i)$ и $T(i, j)$.

Так же на практике в качестве функции разности используется сумма квадратов разностей, кросс-корреляция, либо коэффициенты корреляции (также известной как *нормализованная корреляция*). Последняя применяется в работе [12] для достижения инвариантности по яркости и контрастности.

Обычно для достижения инвариантности относительно различных видов преобразований, шаблон представляют во всех его вариациях, и запускают алгоритм для каждого такого положения шаблона. Так, чтобы алгоритм был устойчив к вращению, необходимо поворачивать шаблон на некоторый

небольшой угол, который будет являться параметром точности алгоритма, и заново искать корреляцию во всех точках изображения для уже повернутого шаблона. Часто для успешного поиска достаточно сделать поиск инвариантным лишь к вращению, масштабированию и сдвигу. При этом выбор функции разности при правильном подборе пороговых величин может обеспечить фотометрический инвариант, как в алгоритме CIRATEFI [12].

Однако, вычислительная сложность прямой реализации устойчивого к преобразованиям корреляционного метода растет экспоненциально от количества степеней свободы шаблона. Именно по этой причине долгое время настоящий метод не рассматривался для практических целей. На сегодняшний день большинство исследований в этой области нацелены на поиск приемлемого способа классифицировать часть преобразований как заведомо ложные, не переводящие шаблон в искомую область. Можно перечислить целый ряд работ по разработке корреляционных методов инвариантных к вращению и масштабированию [13, 14, 15, 16]. Такая растущая популярность так же обусловлена высокой точностью работы этого алгоритма.

Стандартным подходом в уменьшении вычислительной ёмкости корреляционного метода является построение пирамид изображений. Изображение и шаблон представляются в разных масштабах, и алгоритм сперва запускается для самого маленького масштаба. Выбираются области в некоторых окрестностях точек, обладающих максимальной корреляцией. На следующем шаге алгоритм запускается в изображении большего масштаба, но лишь в выделенных областях. Такие зоны поиска именуют *"областями интереса",* также известные в зарубежной литературе как *ROI(Region of Interest)*. В общем случае такой метод способен существенно сократить время вычислений. Однако, когда изначальный размер шаблона относительно мал, как

в случае объектов на аэрофотоснимках, точность вычислений заметно падает, что делает его неприменимым в настоящей работе.

Другой распространённый способ ускорить вычисления заключается в использовании интегрального представления матрицы. Элементы такой матрицы накапливают суммы всех предыдущих элементов, что избавляет от необходимости лишнего пересчёта. Однако, этот подход применим лишь в самой простой реализации корреляционного алгоритма, когда отсутствует вращение и другие виды преобразований. Поэтому в настоящей работе он не рассматривается.

3. Описание метода

В случае аэрофотоснимков, полученных сверхлегким БПЛА, изображения могут быть произвольно ориентированы по азимуту и углу наклона относительно плоскости ландшафта. Это обусловлено непрерывным влиянием погодных условий на БПЛА во время его движения вдоль заданной траектории. Датчик навигации определяет высоту движения планера с погрешностью, и при соотнесении информации автопилота по времени и месту с исследуемыми снимками данные бортового самописца можно использовать лишь для внесения начальных предположений об интервале, в котором следует изменять масштаб шаблона при запуске алгоритма поиска. Существенным искажениям подвергаются зоны фотографии, близкие к краям. Это обусловлено оптическими свойствами линзы, установленной в камере. Принимая во внимание вышесказанное, можно сделать вывод, что для успешного обнаружения шаблона в таких условиях, необходима реализация алгоритма, устойчивого к произвольным аффинным преобразованиям.

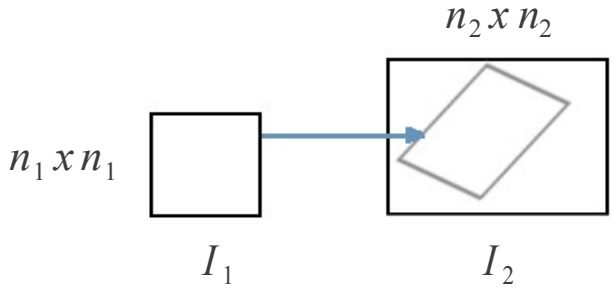


Рис. 1. Аффинное преобразование шаблона в область изображения

Не умаляя общности, будем рассматривать задачу в случае, когда матрицы изображения и шаблона квадратные. Так обозначим через I_1 матрицу шаблона размером $n_1 x\, n_1$. Аналогично определим матрицу изображения $I_2 : n_2 x\, n_2$. Шаблон отображается в участок изображения под действием некоторого

аффинного преобразования, как схематично изображено на *рис. 1*.

Обозначим $\nu(I)$ гладкость изображения, по следующей формуле:

$$\nu = \sum_{p \in I} \max_{q \in N(p)} |I(p) - I(q)| \quad ,$$

которая задаётся суммой по всему изображению максимумов разности пикселей p и восьми соседних пикселей $q \in N(p)$.

Пусть $\Delta_T(I_1, I_2)$ как нормализованная сумма *SAD* будет расстоянием между I_1 и I_2 относительно преобразования T , которое сопоставляет пиксели $p \in I_1$ пикселям в I_2 :

$$\Delta_T(I_1, I_2) = \frac{1}{n_1^2} \sum_{p \in I_1} |I_1(p) - I_2(T(p))| \quad .$$

В случае отображения p за границы I_2 берем разницу $|I_1(p) - I_2(T(p))|$ равной 1 . Пусть имеется некоторое семейство преобразований ψ . Минимум по всем T в данном семействе относительно выбранного расстояния обозначим:

$$\Delta^{\varphi}(I_1, I_2) = \min_{T \in \Psi} \Delta_T(I_1, I_2) \quad .$$

В таком случае задача о поиске максимальной корреляции описывается следующим предложением: *для семейства аффинных преобразований* ψ *и для некоторого положительного параметра точности* $\delta > 0$ *необходимо найти такое* $T^* \in \psi$ *, что выполняется следующее неравенство:*

$$|\Delta(I_1, I_2) - \Delta_{T^*}(I_1, I_2)| < \delta \quad \textit{, где}$$

$\Delta(I_1, I_2)$ *расстояние для оптимального преобразования среди множества всех возможных аффинных преобразований.*

С точки зрения непосредственной реализации алгоритма на ЭВМ в текущей постановке задачи ключевым моментом является правильное построение такого семейства ψ . Оно в первую очередь должно обладать свойством наличия такого T^* . Решение этой задачи было предложено в работе [17], представленной в прошлом году на ежегодной конференции по компьютерному зрению и распознаванию образов. По утверждению самих авторов на сегодняшний день это первый корреляционный метод, гарантирующий поиск с нужным приближением в случае произвольных плоских аффинных преобразований. Для решения задачи выделения объектов на аэрофотоснимках мною был выбран для реализации именно этот алгоритм.

Введем метрику l_∞ на множестве аффинных преобразований:

$$l_\infty(T, T') = \max_{p \in I_1} \|T(p) - T'(p)\|_2 \ ,$$

где $\| \|_2$ - евклидово расстояние в плоскости целевого изображения I_2 . Текущая метрика характеризует насколько далеко друг от друга могут находиться образы точки, полученные применением двух преобразований T и T' .

Для некоторого положительного $\alpha > 0$ семейство $\{T_i\}_{i=1}^{l}$ назовем $\alpha-$*покрытием* , если для всякого аффинного преобразования T в семействе найдется такое T_j , что $l_\infty(T, T_j) = O(\alpha)$. В работе [17] приводится способ построения $(\delta n_1)/2-$*покрытия* , для некоторого параметра точности алгоритма $\delta \in (0,1]$. Построенное покрытие будет иметь размер $\Theta\left(\delta^{-6} \cdot (n_2/n_1)^2\right)$. Реализуемый алгоритм эффективно определяет с высокой вероятностью такое T , что $\Delta_T(I_1, I_2)$ близко к $\Delta(I_1, I_2)$. Вероятность обусловлена функцией, зависящей от δ и общей гладкости шаблона ν .

3.1. Алгоритм FAsT-Match

1. Строим сеть $N_{\delta/2}$, являющуюся $(\delta n_1)/2-$*покрытием* .
2. Для каждого $T \in N_{\delta/2}$ вычисляем $\Delta_T(I_1, I_2)$ с точностью $\delta/2$.
3. Находим T^* : $\Delta_{T^*}(I_1, I_2) = \min_{T \in N_{\delta/2}} \Delta_T(I_1, I_2)$.

На шаге 2 вместо вычисления *SAD* по всем точкам шаблона, мы выбираем лишь некоторые из них. Так, для некоторого положительного $\epsilon > 0$ мы выбираем $m = \Theta(1/\epsilon^2)$ точек $p_1 \dots p_m \in I_1$. Тогда $\Delta_T(I_1, I_2)$ считается по формуле:

$$\Delta_T(I_1, I_2) = \frac{1}{m}\sum_{i=1}^{m} \left| I_1(p_i) - I_2(T(p_i)) \right| \quad .$$

Для того, чтобы новое расстояние Δ_T отличалось от оригинального варианта не более, чем на ϵ с вероятностью $1-\eta$, согласно границе Чернова требуется выбирать количество точек $m = \Theta(\log(1/\eta)/\epsilon^2)$.Всякий раз при вычислении $\Delta_T(I_1, I_2)$ на втором шаге мы добиваемся фотометрического инварианта, проводя нормализацию интенсивности шаблона и изображения на величину их математического ожидания и дисперсии.

Авторы приводят обоснование того факта, что этот алгоритм возвращает преобразование T : $\left|\Delta_T(I_1, I_2) - \Delta(I_1, I_2)\right| \leqslant O\left(\frac{\delta \cdot \nu}{n_1}\right)$ с высокой вероятностью. Для гладких изображений $\nu \approx n_1$. Гладкость является естественным свойством большинства изображений. Более того в случае аэрофотоснимков практически всю площадь изображения занимает равномерный ландшафт местности. Поэтому итоговая оценка удовлетворяет неравенству:

$$\left|\Delta_T(I_{1,}I_2)-\Delta(I_{1,}I_2)\right| \leqslant O(\delta) \ .$$

Сложность алгоритма составляет $\tilde{O}(\frac{1}{\delta^8}\cdot\left(\frac{n_2}{n_1}\right)^2)$. Очевидно, что при малом δ такая сложность не может устраивать. Поэтому авторы предложили способ значительно ускорить вычисления с незначительной потерей точности методом ветвей и границ. При построении начального покрытия берется крупный шаг. После каждой итерации строится новое покрытие с меньшим шагом, но для рассмотрения выбираются лишь те элементы покрытия, для которых найдётся T с достаточно маленьким расстоянием $\Delta_T(I_{1,}I_2)$. Формально процедуру можно описать так:

1. Пусть S^0 полное множество преобразований в покрытии N_{δ_0} (для начального приближения δ_0).
2. Для $i=0$ повторять пока $\delta_i>\delta^*$:

   (а) Запустить алгоритм с точностью δ_i и рассматривать только подмножество $S^i\in N_{\delta_i}$.

   (б) Пусть T_i^{Best} найденное преобразование в S^i .

   (г) Пусть $Q^i = \{q\in S^i : \Delta_q(I_{1,}I_2)-\Delta_{T_i^{Best}}(I_{1,}I_2) < L(\delta_i)\}$.

   (д) Увеличить точность $\delta_{i+1}=\text{fact}\cdot\delta_i$, для некоторого $(0 < \text{fact} < 1)$.

   (е) Пусть $S^{i+1} = \{T\in Net_{\delta_{i+1}} : \exists q\in Q_i$, т.ч. $l_\infty(T,q) < \delta_{i+1}\cdot n_1\}$
3. Вернуть найденную трансформацию T_i^{Best} .

3.2. Алгоритм Ciratefi

В некоторых случаях нам может быть известно, что снимки, на которых проводится обнаружение, достаточно высокого качества, сняты под прямым углом, а матрица камеры не деформирует изображение на краях. В таком случае можно рассматривать не произвольные аффинные преобразования, а ограничиться униформным масштабированием, одинаково меняющим размер шаблона по обеим координатам. В общем случае согласно работе [18] любая матрица аффинного преобразования может быть представлена в виде произведения четырёх матриц:

$$A = Tr\, R_2 S\, R_1 \ ,$$

где Tr, R_i, S соответственно матрицы сдвига, вращения и неуниформного масштабирования. В рассматриваемом случае, мы можем ограничиться лишь матрицей сдвига, одной матрицей вращения и матрицей масштабирования вида:

$$S = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \ .$$

В такой постановке задача решается с высокой точностью с помощью алгоритма CIRATEFI [12] за время $O(n_1^{\,2} n_2^{\,2})$. Алгоритм CIRATEFI состоит из трех каскадных фильтров, которые последовательно исключают из дальнейшего рассмотрения пиксели, у которых нет шансов правильно сопоставить шаблон.

3.2.1. Фильтр выборки по окружностям

Пусть имеется шаблон Q и изображение A . На первом шаге строится множество кандидатов первого класса, прошедших фильтрацию выборки по

окружности. Определим $Cis_B(x, y, r)$ среднее значение пикселей изображения B, расположенных на удалении r от точки (x, y):

$$Cis_B(x, y, r) = \int_0^{2\pi} B(x + r\cos\theta, y + r\sin\theta)\, d\theta \ .$$

Возьмем для рассмотрения n шаблонов различных масштабов $Q_{0,}\ldots, Q_{n-1}$. Выберем l расстояний $r_{0,}\ldots, r_{l-1}$ и определим матрицу C_Q следующим образом:

$$C_Q[i, k] = Cis_{Q_i}(x_{0,} y_{0,} r_k),\ 0 \leqslant i < n,\ 0 \leqslant k < l \ ,$$

где $(x_{0,} y_0)$ центральный пиксель в Q. Это двумерная матрица масштабированных инвариантных к вращению признаков с n строками (масштабы) и l столбцами (радиусы).

Пусть A изображение, в котором ищется шаблон. Тогда построим трехмерное изображение $C_A[x, y, k]$ следующим образом:

$$C_A[x, y, k] = Cis_A(x, y, r_k),\ 0 \leqslant k < l \ \text{ и } \ (x, y) \in domain(A) \ .$$

Найдем максимум корреляции среди всех n масштабов:

$$CisCorr_{A,Q}(x, y) = \max_{i \in [0, n-1]} \left| Corr(C_Q[i], C_A[x, y]) \right| \ .$$

Тогда аргумент, на котором достигнут максимум, будет определять наиболее вероятный масштаб, в котором требуется отображать шаблон на изображение с целью его дальнейшего поиска в точке (x, y). Если $CisCorr_{A,Q}(x, y) > t_1$ для некоторого порогового значения t_1, то точка (x, y) считается кандидатом первого класса. Принцип работы текущего шага алгоритма CIRATEFI иллюстрирует *рис. 2*.

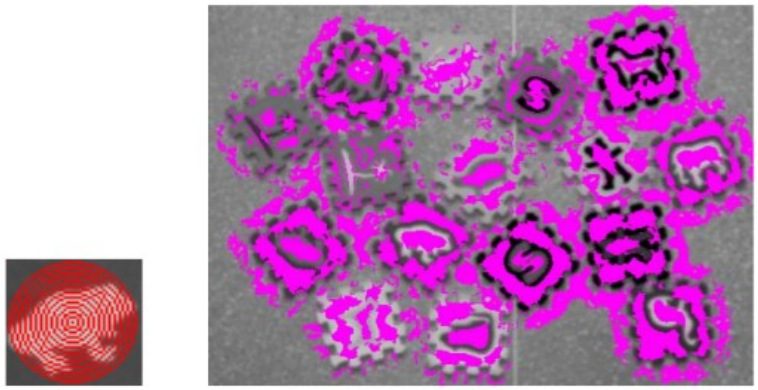


Рис. 2. (а) Фильтр выборки по окружности Cifi. (б) Фиолетовым обозначены точки, получаемые на выходе первого шага алгоритма.

3.2.2. Фильтр выборки по радиусам

Следующий шаг использует проекции A и Q на множества радиус-векторов, чтобы продвинуть точки из первого класса в кандидаты второго уровня. Определим $\mathrm{Ras}_B^{\lambda}(x, y, \alpha)$ как среднее значение интенсивностей пикселей B, расположенных на радиус-векторе с выбранным началом в координатах (x, y) длины λ и углом α к оси OX:

$$\mathrm{Ras}_B^{\lambda}(x, y, \alpha) = \int_0^{\lambda} B(x + t\cos\alpha, y + t\sin\alpha)\,dt \quad .$$

Построим вектор R_Q размерности m следующим образом:

$$R_Q[j] = Ras_Q^{r_{l-1}}(x_{0,}, y_{0,}, \alpha_j),\ 0 \leqslant j < m \quad .$$

Здесь α_i углы с одинаковым шагом, на которые поворачивают радиус-вектор длины $\lambda = r_{l-1}$ (максимальный радиус окружности на первом шаге) с началом в центре шаблона $(x_{0,} y_0)$.

В каждой точке кандидате первого класса (x, y) строятся такие же

вектора для изображения A с длиной радиус-вектора λ равной $s_i r_{l-1}$, где s_i масштаб, найденный на первом шаге. Таким образом, получается трехмерная матрица:

$$R_A[x, y, j] = \mathrm{Ras}_A^{s_i r_{l-1}}(x, y, \alpha_j), \quad 0 \leqslant j < m \ .$$

Далее вычисляется угол, на который необходимо повернуть шаблон, путём поиска максимума корреляции R_Q и R_A :

$$\mathrm{RasCorr}_{A,Q}(x, y) = \max_{j \in [0,\, m-1]} \left| Corr\left(R_A[x, y], \mathrm{cshift}_j(R_Q)\right)\right| \ ,$$

где cshift_j означает циклический сдвиг на j позиций вектора аргумента. Если $\mathrm{RasCorr}_{A,Q}(x, y) > t_2$ для некоторого порогового значения t_2 , то точка (x, y) переводится в кандидаты второго класса.

3.2.3. Фильтр выборки по сопоставлению шаблона с изображением

На последнем шаге происходит классическое сравнение шаблона и изображения по интенсивностям пикселей в точках кандидатах второго уровня. При этом при сопоставлении шаблон масштабируется и поворачивается в соответствии с параметрами, вычисленными на первых двух шагах алгоритма. Точка максимальной корреляции является искомым результатом алгоритма Ciratefi.

4. Реализация

В настоящей работе выполнены реализации в среде параллельных вычислений CUDA обоих алгоритмов FAsT-Match и Ciratefi и проведен сравнительный анализ их работы по тестовым наборам изображений. Основным критерием при выборе технологий являлось время работы. Поэтому в качестве языка реализации был выбран C++. Основную работу по загрузке изображений и наглядному отображению результатов берёт на себя библиотека OpenCV. Ниже следует описание архитектур полученных решений.

На *рис. 3* представлена схема работы алгоритма FAsT-Match. После загрузки изображений, происходит подбор оптимальных параметров. Изображение сглаживается, чтобы избежать больших несоответствий при малых погрешностях в выборе координат точек. Строится множество преобразований с начальным приближением δ_0 . На каждом шаге пока не будет достигнуто нужное приближение $\delta_i = \delta^*$, строится очередное покрытие, элементы которого фильтруются по описанному ранее принципу. В итоге алгоритм по завершении работы выдает аффинное преобразование, давшее лучшую корреляцию шаблона с изображением.

На *рис. 4* представлена схема работы алгоритма Ciratefi. Загруженные изображения так же сглаживаются, и делается автоматическая настройка пороговых значений. Далее происходит каскадная фильтрация, где последовательно строятся множества точек кандидатов первого и второго уровней. Максимальная корреляция достигается на одной из точек второго уровня, и искомое преобразование выдаётся в качестве результата по завершении работы алгоритма.

4.1. FAsT-Match

4.1.1. Схема работы алгоритма

Рис. 3. Схема работы алгоритма FAsT-Match

4.1.2. Особенности реализации

Как уже было сказано в предыдущей главе, согласно [18] мы можем представить матрицу всякого аффинного преобразования в виде произведения четырёх матриц:

$$A = Tr\, R_2 S\, R_1 \quad .$$

Матрица сдвига Tr и матрица масштабирования S задают по две степени свободы. Каждая матрица поворота R_i добавляет ещё одну степень свободы. В итоге любому преобразованию соответствует вектор шести компонент $(t_x, t_y, r_{1,} r_{2,} s_x, s_y)$, где

$$\begin{cases} t_x, t_y \in [-n_{2,} n_2], & с\ шагом\ \Theta(\delta n_1) \\ r_{1,} r_2 \in [0, 2\pi], & с\ шагом\ \Theta(\delta) \\ s_x, s_y \in [1/c, c], & с\ шагом\ \Theta(\delta) \end{cases}$$

Количество шагов для $r_{1,2}$, $s_{x,y}$ равно $\Theta(1/\delta)$, в то время, как $t_{x,y}$ принимают $\Theta(\frac{n_2}{n_1}/\delta)$ различных значений. С учетом ограничений архитектуры CUDA на количество одновременно работающих потоков и максимальный допустимый размер блока, а также принимая во внимание тот факт, что больше всего шагов необходимо сделать для компонент сдвига $t_{x,y}$, было естественным выбрать двумерный блок, где компоненты x и y соответствуют t_x и t_y :

```
#define BLOCK_SIZE 16
...
dim3 dimBlock(BLOCK_SIZE, BLOCK_SIZE);
dim3 dimGrid(2 * I2.cols / (delta * I1.cols),
          2 * I2.rows / (delta * I1.rows));
CUDA_buildAffineSet<<<dimGrid, dimBlock>>>(..., transforms);
```

На первом шаге строится коллекция аффинных преобразований с начальным приближением $\delta = \delta_0$. Для всех векторов в коллекции параллельно вычисляются Δ_T вызовом функции на одномерном гриде:

```
dim3 dimBlock(BLOCK_SIZE * BLOCK_SIZE);
dim3 dimGrid(num_transforms / dimBlock.x);
CUDA_evalCorr<<<dimGrid, dimBlock>>>(..., distances);
```

В полученном массиве расстояний идёт поиск минимума с помощью операции *map reduction*:

```
CUDA_mapReduce<<<dimGrid, dimBlock>>>(distances, ..., best_distance);
```

Если полученное расстояние Δ_{T^*} меньше некоторого выбранного порогового значения, то алгоритм завершает поиск и соответствующее преобразование T^* считается искомым. В противном случае выбирается из начальной коллекции подмножество таких преобразований T_i , которые близки по выбранной метрике к T^* :

$$\Delta_{T_i} - \Delta_{T^*} < L(\delta) \ .$$

Далее следует построение множества преобразований с шагом $\delta_{i+1} = \text{fact} \cdot \delta_i$, для некоторого $0 < \text{fact} < 1$, с последующей фильтрацией по признаку близости к хотя бы одному элементу из выбранного подмножества. Однако прямая реализация существенно увеличила бы вычислительную сложность. Поэтому с незначительной потерей точности можно строить множество преобразований с $\delta = \delta_{i+1}$ непосредственно из элементов подмножества, стохастически возмущая компоненты векторов преобразований слагаемыми, равновероятно принимающими значения $-step, 0, step$.

4.2. Ciratefi

4.2.1. Схема работы алгоритма

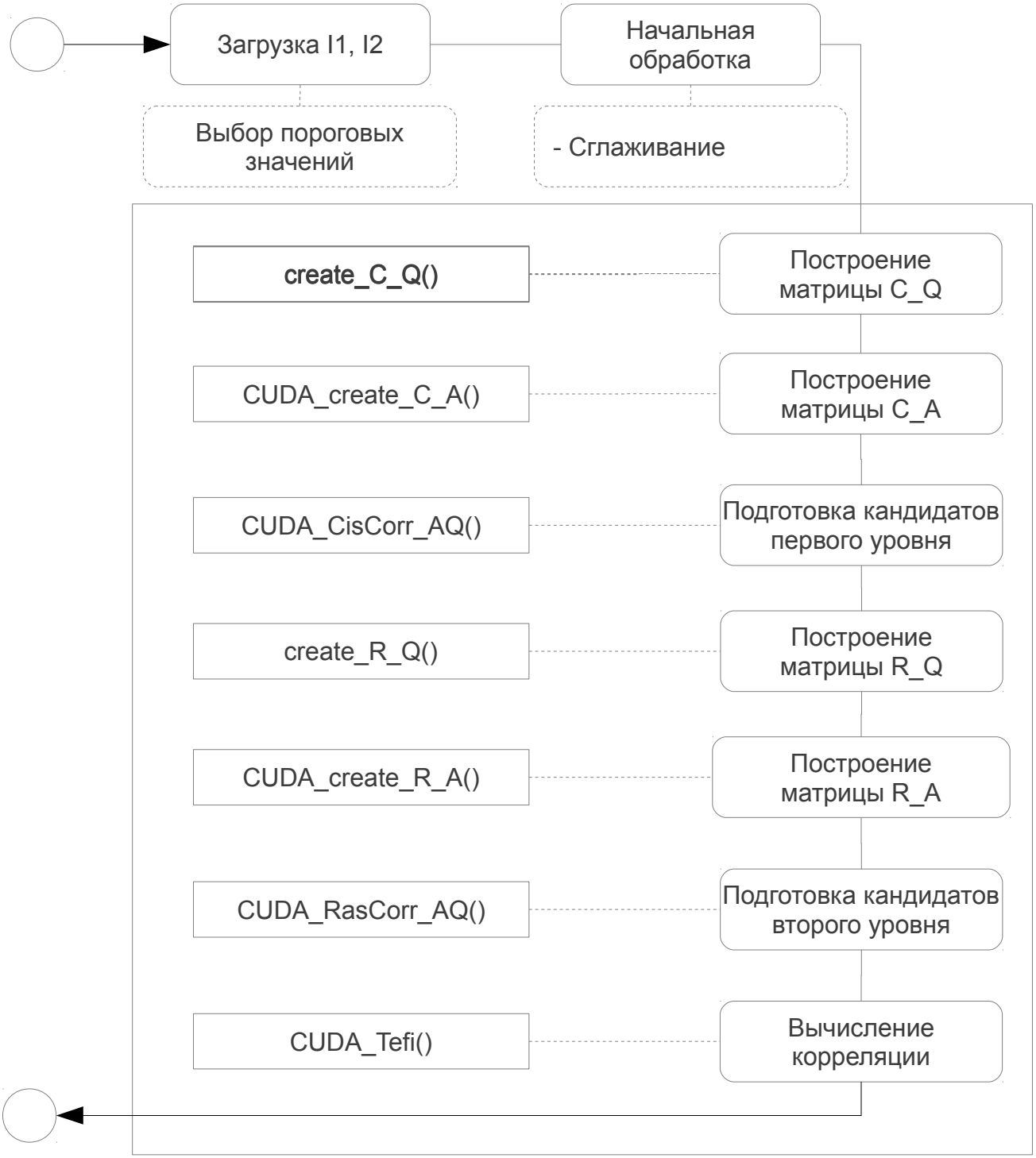


Рис. 4. Схема работы алгоритма Ciratefi

4.2.2. Особенности реализации

Представленный в алгоритме каскадный фильтр на первом уровне вычисляет средние значения интенсивностей пикселей по окружностям, а на втором по отрезками. Как известно, выбор точек некоторой правильной геометрической фигуры, такой как окружности, на дискретном изображении имеет свои особенности. Так, если задавать координаты точек окружности с радиусом R по строгой математической формуле:

$$(R\cos\theta, R\sin\theta),$$

то при угле θ близком к нулю, касательная приближается к вертикальной прямой. В результате получаются заметные пробелы в изображаемой дуге, поскольку многие точки сливаются в одну.

Чтобы получать корректное дискретное изображение окружности, был использован алгоритм *MidpointCircle*, описанный в работе [19]. Точки отрезков строятся по алгоритму *MidpointLine*. Для всех радиусов, для которых строятся окружности в первом фильтре, кэшируются массивы координат точек окружностей. Аналогично для второго фильтра предварительно строится кэш отрезков необходимых длин и наклонов.

В результате профилирования было установлено, что нет необходимости делать параллельную реализацию функций, строящих матрицы C_Q и R_Q. Их суммарное время работы на обычном процессоре занимает сотые доли секунды. Более того, операции по выделению и копированию памяти *host* на *device* и обратно в итоге только бы замедлили вычисления. Аналогичная ситуация при построении кэшей точек окружностей и отрезков.

Основное время у алгоритма занимает построение трехмерных матриц C_A и R_A. Для обработки краевых точек матрица A дополняется полями

на некоторую величину *offset* , зависящую от размеров матрицы Q и максимального масштаба. Качество распараллеливания в CUDA зависит от многих параметров. Среди ключевых критериев можно выделить количество обращений к глобальной памяти и общую загруженность потоков. Поэтому при возможности необходимо как можно более эффективно использовать разделяемую память блоков, обращение к которой в сотни раз быстрее обращений к глобальной памяти.

Как правило, изображение A имеет существенно больше точек, нежели количество доступных для одновременного выполнения потоков. Таким образом, можно обеспечить загруженность, назначив каждому отдельному потоку вычисление одного элемента матрицы C_A (или R_A , если речь идёт о фильтре второго уровня). Естественным образом для их параллельного вычисления выбираются двумерные блоки потоков. Ввиду хаотичности координат точек на окружностях (отрезках), в которых берутся интенсивности пикселей изображения A , не существует общего принципа выделения блоками областей A в разделяемую память. Однако, при детальном подсчёте было обнаружено, что в большинстве случаев разделяемой памяти хватает, чтобы разместить массивы точек окружностей (отрезков). Так, выбранная для экспериментов графическая карта NVidia GTX 560 имеет *compute capability 2.x* и располагает максимальным объемом разделяемой памяти в 48Кб. Координата точки описывается структурой размера 8 байтов:

```
struct Point2D { int x, y }
```

Окружность радиуса 128, построенная алгоритмом *MidpointCircle* с пиксельным шагом 2 имеет 368 точек. Совокупность радиус-векторов длины 128, построенная алгоритмом *MidpointLine* с пиксельным шагом 2 и шагом угла

$\pi/18$ имеет 4309 точек. Эти параметры позволяют проводить вычисления Ciratefi с достаточно высокой точностью для довольно крупных шаблонов. Как видно, разделяемой памяти хватает, чтобы хранить координаты точек. При вызове функции третьим управляющим параметром динамически указывается количество разделяемой памяти. В текущем случае точная величина не принципиальна, и можно задать её максимальный объём:

```
CUDA_create_C_A<<<dimGrid, dimBlock, MAX_SHARED_MEMORY>>>(A,
    circles, ..., C_A)
```

В теле процедуры задаётся цикл по окружностям всех радиусов, на которых считается средняя интенсивность. В каждой итерации координаты точек считываются в разделяемую память, которая используется каждым потоком текущего блока для вычисления средней интенсивности по этим координатам.

При вычислении $\mathrm{CisCorr}_{A,Q}$ и $\mathrm{RasCorr}_{A,Q}$ в разделяемую память полностью копируются соответственно матрицы C_Q и R_Q . На последнем шаге элементы шаблона Q целиком не помещаются в разделяемую память. Поэтому задаётся вложенный цикл по y и x следующим образом:

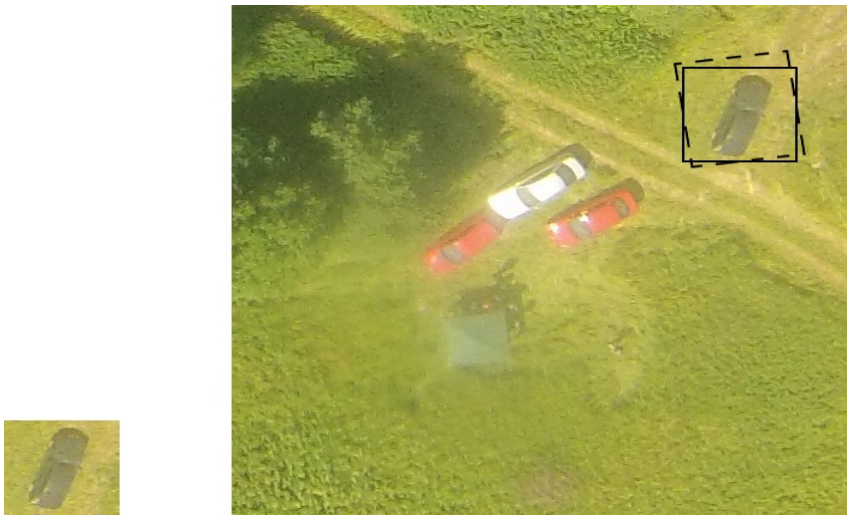
```
for (int y = 0; y < Q.height / BLOCK_SIZE; ++y) {
  for (int x = 0; x < Q.width / BLOCK_SIZE; ++x) {
    __shared__ double Qs[BLOCK_SIZE][BLOCK_SIZE];
    Matrix2d Qsub = _cu_getSubMatrix(Q, y, x);
    . . .
    Qs[threadIdx.y][threadIdx.x] =
        _cu_getElement(Qsub, threadIdx.y, threadIdx.x);
    __syncthreads();
```

Далее внутри цикла накапливается частичная корреляция участков Q и соответствующих участков изображения A . Результирующий массив корреляций мал и не требует проведения операции *map reduce*.

5. Апробация

В настоящей работе проводится сравнение алгоритмов в ходе двух экспериментов. В первом сценарии каждый шаблон берется непосредственно с того же изображения, где будет осуществляться его поиск. Во втором эксперименте шаблон берется с одной фотографии и сопоставляется с другими снимками, которые были сделаны БПЛА за короткий интервал времени относительно первого.

В качестве метрики точности работы алгоритмов была выбрана *ошибка пересечения (overlap error),* которая характеризует отклонение полученной области сопоставления шаблона от оптимальной, как представлено на *рис. 5*.



(а) (б)

Рис. 5. (а) Шаблон (б) Сплошной линией обозначена оптимальная область отображения. Пунктирной полученное преобразование на выходе алгоритма.

Согласно Mikoalyciz [20] ошибка пересечения определяется по следующей формуле:

$$1 - \frac{|\text{Пересечение областей}|}{|\text{Объединение областей}|} .$$

Для первого эксперимента было выбрано 10 снимков. На каждом был отдельно обозначен объект поиска и вычислено оптимальное преобразование. Далее каждый снимок был произвольным образом отмасштабирован и повернут (в последствии также обрезан до единых для всех экспериментов размеров 512 x 512 пикселей). На них шаблон выбирался в произвольной области размерами от 10% до 50% относительно размера исходного изображения. Эксперименты осуществлялись на процессоре Intel Core i5-3450 и графической карте Nvidia GTX 560. Результаты работ алгоритмов в первом эксперименте на CPU и CUDA приведены на *рис.6*.

	10 приг. снимков	50%	20%	10%
Overlap. error FAsT-M	4.5%	7.1%	9.3%	14.2%
Overlap. error Ciratefi	5.2%	12.6%	12.1%	10.3%
Время FAsT-M CPU	54c.	38c.	47c.	52c.
Время Ciratefi CPU	27c.	37c.	31c.	26c.
Время FAsT-M GPU	6.3c.	2.1c.	5.4c.	6.5c.
Время Ciratefi GPU	2.5c.	3.8c.	3.2c	2.7c.

Рис. 6. Сводная таблица результатов первого эксперимента

Во втором эксперименте было подготовлено три коллекции снимков. Задача этого сценария продемонстрировать результаты работы алгоритмов в условиях, приближенных к реальным. В первой коллекции в качестве шаблона обнаружения был выбран шатёр. Во второй легковая машина на стоянке. В третьей крайнее дерево из лесополосы. В каждой коллекции было выбрано по две фотографии, откуда в дальнейшем были взяты изображения шаблонов. Далее шел поиск корреляции этих шаблонов со всеми остальными снимками коллекций. Результаты работы представлены на *рис. 7*.

	1 колл.	2 колл.	3 колл.
Overlap. error FAsT-M	3.2%	3.6%	2.4%
Overlap. error Ciratefi	3.6%	4.4%	7.0%
Время FAsT-M CPU	48c.	44c.	49c.
Время Ciratefi CPU	26c.	31c.	29c.
Время FAsT-M GPU	5.9c.	2.5c.	5.8c.
Время Ciratefi GPU	2.3c.	3.3c.	3.6c.

Рис. 7. Сводная таблица результатов второго эксперимента

В первом эксперименте при произвольном построении изображений в качестве шаблона поиска часто выбирались области ландшафта, что обуславливает рост ошибки вычислений. Как видно время работы алгоритма FAsT-Match зависит обратно пропорционально размеру шаблона. Ciratefi, наоборот, делает меньше вычислений в случае, когда размер шаблона относительно невелик.

Точности алгоритмов можно настраивать выбором различных пороговых значений. В проведенных экспериментах начальное приближение алгоритма FAsT-Match $\delta_0 = 0.25$. Пороговые значения для алгоритма Ciratefi соответственно $t_1 = 0.95$, $t_2 = 0.9$, $t_3 = 0.75$. На *рис. 8* представлены некоторые результаты работы алгоритма FAsT-Match и Ciratefi.

(а) (б)

Рис. 8. (а) Результаты работы алгоритма FasT-Match (б) Ciratefi

6. Заключение

Результаты

В рамках настоящей работы были получены следующие результаты:

1. Изучены существующие подходы в области анализа изображений и сделана оценка их применимости к задаче выделения объектов на аэрофотоснимках.
2. В ходе анализа, были выбраны для реализации два современных алгоритма FAsT-Match и Ciratefi.
3. Выполнена их эффективная реализация в среде параллельного выполнения CUDA.
4. Реализованные подходы были апробированы на конкретных снимках, сделанных в ходе съёмки БПЛА, и проведен сравнительный анализ их работы.

Список литературы

1. Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1):34-58.

2. Torralba, A. (2007). Classifier-based methods. Recognizing and Learning Object Categories. In CVPR 2007 Short Course on http://people.csail.mit.edu/torralba/shortCourseRLOC/.

3. Hastie, T., Tibshirani, R., and Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, New York.

4. Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005), pp. 886–893, San Diego, CA.

5. H. P. A. Nobre and Hae Yong Kim, "Automatic VHDL Generation for Solving Rotation and Scale-Invariant Template Matching in FPGA," V Southern Programmable Logic Conference, São Carlos, pp.21-26, 2009.

6. Brice, C. R. and Fennema, C. L. (1970). Scene analysis using regions. Artificial Intelligence, 1(3-4):205–226.

7. Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6):679–698.

8. K. Mikolajczyk, A performance evaluation of local descriptors, IEEE Trans. on Pattern Analysis and Machine Intelligence , vol. 27, n. 10, 2005, pp. 1615-1630.

9. D.G. Lowe. Distinctive image features from scale-invariant key- points. IJCV, 60(2):91–110, 2004.

10. Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool. SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359, 2008.

11. J.M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences, 2(2):438–469, 2009.

12. H. Y. Kim and S. A. Araújo, "Grayscale Template-Matching Invariant to Rotation, Scale,

Translation, Brightness and Contrast," IEEE Pacific-Rim Symposium on Image and Video Technology, Lecture Notes in Computer Science, vol. 4872, pp. 100-113, 2007.

13. Li, J.H., Pan, Q., Cui, P.L., Zhang, H.C., Cheng, Y.M.: Image recognition based on invariant moment in the projection space. In: Int. Conf. Machine Learning and Cybernetics, Shangai, Vol. 6 (Aug. 2004) 3606-3610.

14. Flusser, J., Suk, T.: Rotation moment invariants for recognition of symmetric objects. IEEE T. Image Processing 15(12) (Dec. 2006) 3784-3790.

15. Dionisio, C.R.P., Kim, H.Y.: A supervised shape classification technique invariant under rotation and scaling. In: Int. Telecommunications Symposium (2002) 533-537.

16. Tao, Y., Ioerger T.R., Tang, Y.Y.: Extraction of rotation invariant signature based on fractal geometry. In: IEEE Int. Conf. Image Processing, vol. 1 (2001) 1090-1093.

17. Korman S., Reichman D., Tsur G. and Avidan S., "FAsT-Match: Fast Affine Template Matching", CVPR2013.

18. R. Hartley and A. Zisserman. Multiple view geometry in computer

vision. Cambridge university press, 2008.

19. James D. Foley e.a.: Computer Graphics: Principles and Practice in C, pp. 86-87

20. K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. IJCV, 65(1):43–72, 2005.