**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа «Прикладная математика и информатика»

**Отчет о программном проекте**

На тему: **Программа обнаружения изменений на спутниковых данных** (Program for change detection algorithms on Satellite data)

**Выполнил:**

студент группы БПМИ 1911 Горбачев Ринат



Подпись И.О. Фамилия

01.06.2021

Дата

**Принял:**

руководитель проекта Родригес Залепинос Рамон Антонио

Имя, Отчество, Фамилия

Доцент

Должность, ученое звание

Факультет компьютерных наук / Департамент программной инженерии

Место работы (Компания или подразделение НИУ ВШЭ)

|  |  |  |
| --- | --- | --- |
| Дата проверки 01.06.2021 | 8 | *Родригес* |
|  | Оценка | Подпись |
|  | (по 10-тибалльной шкале) |  |

**Москва 2021**

Содержание

[Основные термины и определения 3](#_Toc73538319)

[Обзор и сравнительный анализ источников и аналогов 5](#_Toc73538320)

[Методы и алгоритмы 9](#_Toc73538321)

[Реализация основных алгоритмов 13](#_Toc73538322)

[Приложение А 13](#_Toc73538323)

[Приложение Б 13](#_Toc73538324)

[Приложение С 15](#_Toc73538325)

# **Основные термины и определения**

1. **ГИС** - географическая информационная система
2. **ДЗ** - дистанционное зондирование.
3. **Сцена** – географический снимок разных слоев.
4. **Скрипт (сценарий)** -это последовательность действий, описанных с помощью скриптового языка программирования (Python и др.) для автоматического выполнения определенных задач.
5. **Спектральные каналы** – это диапазоны длин волн электромагнитного спектра. Спутники и прочие устройства получения снимков комбинируют эти каналы для наблюдения определённых объектов, которые при таких длинах волн лучше видно.
6. **Change detection (CD) -** обнаружение изменений (ОИ)
7. **Principal-сomponent** **analsis (PCA) –** анализ главных компонент**.**
8. **Remote sensing (RS)** - ДЗ Земли (ДЗЗ).
9. **Normalized Difference Vegetation Index (NDVI)** - нормализованный относительный индекс растительности - простой количественный показатель количества фотосинтетически активной биомассы (обычно называемый вегетационным индексом).
10. **Supervised Classification** - методы используются для создания карты, в которой каждый пиксель назначается классу на основе его мультиспектрального состава. Классы определяются на основе спектрального состава областей обучения, определяемых пользователем.
11. **Modified Soil-adjusted Vegetation Index (MSAVI)** - индекс растительности с поправкой на почву, который направлен на устранение некоторых ограничений NDVI.
12. **Normalized Difference of Time Series (NDTS)** - нормализованная разница временных рядов.
13. **GDAL (Geospatial Data Abstraction Library)** - библиотека для чтения и записи растровых и векторных геопространственных форматов данных.

Введение

Обнаружение изменений поверхности земли является актуальной задачей начиная с 19 века, когда была сделанна первая аэрофотосъемка земли. Впоследстии развитие в данной области было тестно связано с военной техникой и стратегическим преимуществом полученным с помошью ДЗЗ во времена войн. Сейчас же, данные полученные с помощью ОИ широко используются для обнаружения пожаров, вырубки лесов, роста городов, а так же могут применяться в разных областях науки: Географии, истории, экономики. Основным источником получаемых данных являются: Спутниковые снимки, аэрофотоснимки, различные карты.

В контексте дистанционного зондирования обнаружение изменений относится к процессу выявления различий в состоянии особенностей земли путем наблюдения за ними в разное время. Этот процесс может выполняться вручную или с помощью программного обеспечения дистанционного зондирования. Ручная интерпретация изменений по спутниковым изображениям или аэрофотоснимкам предполагает, что наблюдатель или аналитик определяет интересующие области и сравнивает их между изображениями, полученными за две даты. Это может быть выполнено либо на экране (например, в ГИС), либо на бумаге. Интерпретация изображений вручную хорошо работает при оценке изменений между отдельными классами (например, проемы леса, карты землепользования и земного покрова) или когда изменения значительны (например, тяжелые механизированные маневры, воздействия инженерной подготовки). Интерпретация изображений вручную также возможна при попытке определить изменение с помощью изображения или фотографии из разных источников (например, сравнение исторических аэрофотоснимков с текущими спутниковыми снимками).  
Наш проект поможет обнаружить различные изменения на спутниковых данных[1], например:

* Появление новых жилых комплексов
* Изменение территории и ландшафта вызванные катаклизмами
* Появление различных сооружений

В ходе работы мы будем использовать: Алгоритмы, обнаружения изменений, ГИС платформы для получения сцен, интерфейс для визуализации, сервер и возможно автоматизированное приложение.

Моей задачей в командном проекте является анализ и сравнение возможных аналогов и платформ, которые могут получать и работать со снимками нашей планеты. А так же поиск ,анализ и написание алгоритмов ОИ.

# **Обзор и сравнительный анализ источников и аналогов**

Важным фактором в применение методов обнаружения изменений является выбор правильных сцен(аэроснимков). Сцены должны быть получены с одного и того же сенсора при одинаковом угловом разрешении, освещении, в одно время года с минимальным процентом облаков (<20%).

Внедрение ОИ на основе ДЗ может быть выполнено с помощью большинства коммерчески доступного программного обеспечения ГИС / ДЗ. Специализированные

приложения для обработки изображений могут значительно облегчить работу.

Ниже приведена таблица сравнение самых популярных существующих аналогов.

Таблица 1 – Различные платформы для получения сцен(аэроснимков)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Платформа/ресурс | Описание | Преимущества | Недостатки |
| 1 | **ENVI [8]** | Предоставляет автоматизированные рабочие процессы для многих распространенных задач обработки изображений, включая ОИ | 1. Специальная команда разработчиков постоянно выпускает обновления  2. Удобный и современный интерфейc | 1. Полностью на английском языке  2. Платное приложение  3. Недоступно без VPN для пользователей СНГ |
| 2 | **ERDAS [9]** | Выполняет ОИ с помощью стандартных инструментов. | 1. Идеален для новичков  2. Интуитивно понятный | 1. Работает в основном со снимками |
| 3 | **IDRISI Land Change Modeler [13]** | Автоматизированный и простой в использовании инструмент для реализации методов ОИ | 1. Быстрое создание графиков и карт изменения земель.  2. Выявление основных тенденций сложных изменений земель | 1. Платное приложение с продлеванием лицензии.  2. Встроенные дополнительные покупки |
| 4 | **Feature Analyst [10]** | Программное обеспечение для ОИ, включает в себя тонкие настройки масштабирования, формы для определения объектов ландшафта. Работает лучше, чем традиционные методы классификации для отображения дискретных объектов (например, дорог, троп, нарушенных участков) | 1. Обнаружение изменений для конкретного объекта  2. Улучшение результатов извлечения признаков с помощью итеративного иерархического процесса обучения | 1.Операционная система только Windows (не ниже 7 версии)  2. Полностью на английском языке |
| 5 | **ArcGIS[2]** | Самый распространенный программный пакет ГИС - может использоваться для ручного ОИ | 1. Стандартная среда разработчика ГИС-приложений  2. Открытая платформа  3. Поддержка множества языков программирования | 1. Платное приложение  2. Ограниченная поддержка разработчиков |
| 6 | **Google Earth Engine[14]** | Облако-ориентированная кроссплатформенная среда, которая позволяет визуализировать и анализировать сцены нашей планеты. | 1. Позволяет использовать python для работы  2. Содержит обширную базу данных  3. Не требует много времени для запуска и скачивания сцен.  4. Можно скачивать прямо в облако гугл-драйв экономя место на компьютере  5. Позволяет работать с сервером на прямую, что полезно в будущей реализации приложения | 1. Требует определенного уровня понимания работы с python  2. Нужно изучить документацию по работе с изображениями.  3. Полностью на английском  4. Иногда зависает |
| 7 | **Earthexplorer USGS[7]** | Эта больша организация занимающаяся наукой связанной с анализом и слежением сцен земли | 1. Есть все самые популярные настройки для каллибровки и получения изображения | 1. Требует длительного времени подтверждения,чтобы скачать сцены в хорошем качестве  2. Нельзя скачать только избранные сцены, обычно приходит архив~ 1-2GB |

В ходе работы с различными ресурсами наиболее удобными оказались: (7) Earthexplorer.usgs.gov.[7], (6) Google engine[12].

Оба ресурса бесплатны и требуют наличие регистрации, сцены выбираются путем разметки нужной области на карте или заданием координат. (7) обладает интуитивно понятным интерфесом, но для получения сцен с хорошего сенсора, например Landsat-8 [15], нужно подтверждение, которое обычно занимает 1-2 часа, к тому же размер скачиваемых данных превышает 1гб, что не всегда удобно. Для работы с (6) нужно знание скрипта для отправки запросов и получения нужной информации. Можно очень точно указать параметры сцены : сенсор, спектральный канал, погодные условия итд. Данные могут быть получены за 5-10 минут, причем их размер можно масштабировать.

Так же, что важно, скриптинговая система в (6) позволяет в дальнейшем масштабировать проект: добавить интерфейс, запустить скрипт на удаленном сервере, добавить консоль для работы с запросами клиентов и так далее. Поэтому в дальнейшем буду использовать именно ее.

После получения сцен следующим этапом является применение алгоритма ОИ.

Таблица 2 – Основные алгоритмы ОИ, пиксельно-ориентированные [3]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Техника | Суб-класс | Подход | Преимущества | Недостатки | Пример |
| Пиксельно -ориентированная | Прямое сравнение | Разность параметров | 1. Просто  2. Легко  интерпретировать результаты | 1. Большая погрешность  2. Почти не позволяет классифицировать результат  3. Двоичный результат | Изменения городского земного покрова на окраинах городов по снимкам |
|  | Преобразование изображения | Индексный(Veg,NDTS etc) | Снижает влияние топографических эффектов и освещения. | 1. Случайный или когерентный шум  2. Двоичный результат | Определение типа лесозаготовок |
|  |  | Анализ главных компонентов (PCA) | 1.Снижение избыточности данных  2.Подчеркивает различную информацию в производных компонентах | 1. Зависит от качества сцены  2. Не делает различий между типами изменений (двоичный результат) | 1. Ущерб от пожаров  2. Анализ изменений в землепользовании |

Таблица 3 - Основные алгоритмы ОИ, объектно-ориентированные [3]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Техника | Суб-класс | Подход | Преимущества | Недостатки | Пример |
| Объектно -ориентированная | На основе прямого сравнения объектов | Объекты, найденные на одной сцене ищуться на второй | 1. Мы сами классифицируем объекты  2.Простая интеграция с ГИС | 1. Сложность поиска пространственно соответствующих объектов на разновременных изображениях  2. Сложно задать порог точности, так как какждого обьекта он свой | Обнаружение изменений ландшафта горной породы |
|  | Классификация объектов на основе сравнения | Каждая сцена обрабатывается отдельно и далее сравниваются результаты обработки | 1. Доступно для всех обьектов на сценах  2. Есть учет классификации объектов  3. Полная матрица изменений как результат | 1. При поиске объектов, извлеченных из одного изображения на втором изображении, ошибка местоположения может привести к неверным результатам изменения  2. Зависит от точности сегментации | Обнаружение появлений различных объектов и их классификация |

Понятно, что область ОИ намного обширнее и я привел лишь часть методов. Более продвинутые методы основаны на машинном обучение и нейронных сетях. Пока я остановлюсь на более простом. Реализация некоторых алгоритмов из рассмотренных в таблице приведена дальше.

# **Методы и алгоритмы**

После получения сцен[Приложение А] стоит вопрос о выборе самого алгоритма CD. Основной язык реализации – python3. Принципиально есть два метода: (1) Пиксельно-ориентированный (pixel-based) и (2) обьектно-ориентированный (object-based).

(1) – классический метод использующий пиксели сцены в качестве основной единицы анализа. Внутри этого метода есть различные подходы – алгоритмы. Среди них я выбрал : (1.1) индексный метод на основе NDTS и (1.2) метод PCA основанный на работе с векторами.

Перед началом применения самих алгоритмов или в их процессе я перевожу анализируемые сцены в серый диапазон, так как значения серого в различных диапазонах кодируют изменения яркости *(x, y)* в интервале длин волн *∆λ* из-за солнечного света, отраженного от некоторой точке *(x, y)* на поверхности Земли, или из-за теплового излучения этой поверхности на которую фокусируется датчики сенсора. Игнорируя все эффекты поглощения и рассеяния промежуточной атмосферы, яркость на датчике, доступная для измерения отраженного солнечного света от горизонтальной *Ламбертовской* поверхности, то есть поверхности, которая равномерно рассеивает отраженное излучение во всех направлениях.

Все это выражается формулой

*L∆λ(x, y) = E∆λ · cos θz · R∆λ(x, y) / π.*  (1)

*E∆λ* - средняя спектральная освещенность Солнца в спектральной полосе *∆λ, θz* - зенитный угол Солнца*, R∆λ (x, y)* - коэффициент отражения поверхности в координатах *(x, y),* число от 0 до 1, а *π* соответствует верхней полусфере телесного угла.

Я не буду вдаваться в подробности более детально, пока это не нужно. Главное понятно, что сцены будут в сером диапазоне. Кстати, имея значения пикселей сцены в rgb, в серый ее можно трансформировать функцией Grayscale из библиотеки PIL или так

*gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b*

Заметим, что переводя снимки в серый диапазон мы теряем часть информации и это так, но этого требует подход (1). С этой проблемой борется подход (2), который рассматривает раразличные обьекты на сцене, классифицирует их с использованием пространственной и спектральной информации, но он заметно сложнее.

В качестве примера работы алгоритма (1.1)[Приложение B] были взяты две сцены одной местности за 2015[рис.1] и 2018[рис.2] годы. После применения алгоритма получен результат[рис.3] где белые точки сигнализируют о наличии изменений за данный промежуток времени.



Рисунок 1. 2015 год Рисунок 2. 2018 год

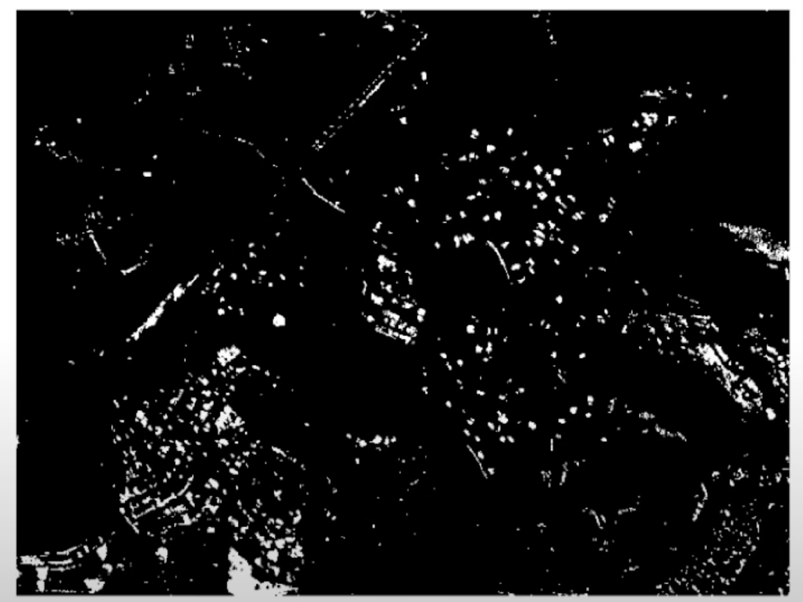


Рисунок 3. Результат алгоритма

Для алгоритма (2)[Приложение С] нужно вспомнить базовые понятия линейной алгебры.

Наши сцены (изображения), например как на рисунке 1 и 2, могут быть представлены как трехмерный массив значений шкалы серого или интенсивности пикселей.

(1.1)

где c - количество столбцов пикселей (также называемых выборками), а r - количество строк пикселей. Индекс k обозначает спектральную полосу, коих всего N.

Пиксель мы будет представлять как

|  |  |
| --- | --- |
|  | (1.2) |

что можно интерпритировать как вектор-столбец спектральных интенсивностей или значений шкалы серого в позиции изображения (i, j).

Длина *Евклидовой* нормы вектора x

(1.3)

При анализе изображений часто необходимо найти собственные значения вектора. В простейшем случае, который будет интересовать нас в первую очередь, нахождение собственных векторов u и собственных значений λ, удовлетворяющих матричному уравнению

*Au = λu*, (1.4)

где A является как симметричной и положительно определенной матрицей. Геометрически мы ищем специальные векторы u, которые при умножении матрицы на A изменяют не более чем свой знак и длину, но не направление, то есть собственные векторы A. Уравнение (1.4) может быть записано эквивалентно как

(A - λI) u = 0 (1.5)

Для нетривиальных решений, то есть если u 0 имеем

|A − λI| = 0. (1.6)

Это хорошо известное характеристическое уравнения матрицы А, решая его получаем собственные значения матрицы А, . Мы знаем ,что собственные значения ортогональны по определению, к тому же уравнение(1.4) можно домнажать на любую константу 0 слева и справа, поэтому можно сделать так, чтобы ∥)∥ = ∥∥ = 1

(1.7)

Следовательно, мы можем составить матрицу *U* из собственных значений, то есть

,) = (1.8)

Здесь мы переходим к диагонализации матрицы А

Λ = (1.9)

Зная, что , домножим слева на *U* и получим

(1.10)

Мы будем разлогать наши сцены на набор массивов пикселей и состовлять из них матрицы. Для анализа и применения алгоритмов улучшения нам понадобится знание сингулярного разложения SVD. Домножим (1.10) на справа получим

*A = UΛU⊤* (1.11)

Это говорит о том, что любая симметричная матрица A может быть разложена на произведение ортонормированной матрицы *U* на диагональную матрицу , диагональные элементы которой являются собственными значениями A, умноженными на транспонированнную *U*.

Алгоритм прилагается в Приложение С.

# 

# **Реализация основных алгоритмов**

## **Приложение А**

var s2 = ee.ImageCollection("LANDSAT/LO08/C01/T1\_RT") //задаем сенсор

.filterBounds(geometry) // задаем местность на карте(фигуру)

.**filter**(ee.**Filter**.lessThanOrEquals('CLOUDY\_PIXEL\_PERCENTAGE', 10)) //облачность

.**filter**(ee.**Filter**.date('2020-05-01', '2021-05-31')) // нужные даты

var clip\_all = function(img){ // все подходящие сцены совподающие с фигурой

**return** img.clip(geometry)

}

var s2\_clip = s2.**map**(clip\_all)

var s2\_list = s2\_clip.toList(s2\_clip.size())

var n = s2\_list.size().getInfo()

**for** (var i = 0; i < n; i++){

var img = ee.Image(s2\_list.get(i))

// нужные нам каналы В2-синий, B3-зеленый, B4-красный , то есть RGB

var rgb = img.select('B2','B3', 'B4').**float**()

// дата изображения идет в название сцены, при получении.

var date = img.date().format('yyyyMMdd').getInfo()

// ограничиваем rgb канал, т.к значиний очень много, но основную инф. получили

**Map**.addLayer(img, {bands: ['B4', 'B3', 'B2'], **max**: 5048}, 'Image\_'+date)

// Сцены можно сохранить себе на гугл драйв

Export.image.toDrive({

image: rgb,

description: 'LANDSAT8\_RGB'+ date,

scale: 10,

region: geometry,

folder: 'Paris' })}

## **Приложение Б**

**from** gdalconst **import** \*

# необходимо для создания графиков

**import** matplotlib.pyplot **as** plt

# необходимо для работы с массивами

**import** numpy **as** np

# необходимо для вычисления mode массивов

**from** scipy **import** stats

# файлы со сценами

filename\_t1 = "Sentinel2\_RGB20210331.tif"

filename\_t2 = "Sentinel2\_RGB20180418.tif"

# параметры изображения

rows = 500

columns = 700

figure\_border = 25

# Загружаю datasets

dataset\_t1 = gdal.**Open**(filename\_t1, GA\_ReadOnly)

dataset\_t2 = gdal.**Open**(filename\_t2, GA\_ReadOnly)

# Получаем информацию параметра B1 для обоих снимков

### NDVI

ndvi\_t1 = dataset\_t1.GetRasterBand(1)

ndvi\_t2 = dataset\_t2.GetRasterBand(1)

# Построим гистограммы для обоих снимков bands 1

plt.figure(figsize=(10, 8))

plt.plot(ndvi\_t1.GetHistogram(), label="$I\_{t\_1}$")

plt.plot(ndvi\_t2.GetHistogram(), label="$I\_{t\_2}$")

plt.legend()

plt.grid(b=True, color='gray', linestyle='--', linewidth=0.5);

# plt.savefig("histograms\_t1xt2.png", format='png', dpi=200)

plt.show()

plt.close()

# Считаем индекс NDTS

array\_t1 = ndvi\_t1.ReadAsArray()[0:rows, 0:columns].astype(**float**)

array\_t2 = ndvi\_t2.ReadAsArray()[0:rows, 0:columns].astype(**float**)

array\_ndts = (array\_t2 - array\_t1) / (array\_t2 + array\_t1)

# Строим первичный NDTS

output\_fig, (ndts\_ax, histogram\_ax) = plt.subplots(figsize=(10, 3), ncols=2)

ndts\_ax.imshow(array\_ndts, cmap="gray")

ndts\_ax.set\_xlim([0 - figure\_border, columns + figure\_border])

ndts\_ax.set\_ylim([rows + figure\_border, 0 - figure\_border])

histogram\_ax.hist(array\_ndts.ravel(), bins=200, **range**=(-1.0, 1.0))

histogram\_ax.grid(b=True, color='gray', linestyle='--', linewidth=0.5);

# output\_fig.savefig("ndts.png", format='png', dpi=200)

plt.show()

plt.close()

# Применяю алгоритм улучшения

# квадрат NDTS

array\_square\_ndts = array\_ndts \* array\_ndts

# Строим квадрат NDTS

output\_fig, (square\_ndts\_ax, histogram\_ax) = plt.subplots(figsize=(10, 3), ncols=2)

square\_ndts\_ax.imshow(array\_square\_ndts, cmap="gray")

square\_ndts\_ax.set\_xlim([0 - figure\_border, columns + figure\_border])

square\_ndts\_ax.set\_ylim([rows + figure\_border, 0 - figure\_border])

histogram\_ax.hist(array\_square\_ndts.ravel(), bins=200, **range**=(-0.25, 0.25))

histogram\_ax.grid(b=True, color='gray', linestyle='--', linewidth=0.5);

# output\_fig.savefig("square\_ndts.png", format='png', dpi=200)

plt.show()

plt.close()

# Метод threshold для выделения результатов в обнаружении изменений

threshold = 0.1

threshold\_array\_square\_ndts = array\_square\_ndts > threshold

# Построим диаграмму с первичным обнаружением изменений

output\_fig, threshold\_ax = plt.subplots(figsize=(10, 8), ncols=1)

threshold\_ax.imshow(threshold\_array\_square\_ndts, cmap="gray")

threshold\_ax.set\_xlim([0 - figure\_border, columns + figure\_border])

threshold\_ax.set\_ylim([rows + figure\_border, 0 - figure\_border])

# output\_fig.savefig("threshold\_square\_ndts.png", format='png', dpi=200)

plt.show()

plt.close()

# Применим алгорит 3x3 mode для уменьшение шумов

mode\_threshold\_array\_square\_ndts = threshold\_array\_square\_ndts

**for** i **in** **range**(1,rows):

**for** j **in** **range**(1,columns):

values = np.array(threshold\_array\_square\_ndts[i-1:i+2, j-1:j+2])

mode\_threshold\_array\_square\_ndts[i, j] = stats.mode(values.ravel())[0][0]

# РЕЗУЛЬТАТ

# создадим фигуру уже отфильтрованного обнаружения изменений

output\_fig, threshold\_ax = plt.subplots(figsize=(10, 8), ncols=1)

threshold\_ax.imshow(mode\_threshold\_array\_square\_ndts, cmap="gray")

threshold\_ax.set\_xlim([0 - figure\_border, columns + figure\_border])

threshold\_ax.set\_ylim([rows + figure\_border, 0 - figure\_border])

# output\_fig.savefig("mode\_threshold\_square\_ndts.png", format='png', dpi=200)

plt.show()

plt.close()

## **Приложение С**

**import** numpy **as** np

**from** osgeo **import** gdal

**import** sys

**from** osgeo.gdalconst **import** GA\_ReadOnly,GDT\_Float32

**def** pca(infile,outfile):

gdal.AllRegister()

inDataset = gdal.**Open**(infile,GA\_ReadOnly)

cols = inDataset.RasterXSize

rows = inDataset.RasterYSize

bands = inDataset.RasterCount

# матрица ‘data’

G = np.zeros((rows\*cols,bands))

k=0

**for** b **in** **range**(bands):

band = inDataset.GetRasterBand(b+1)

mp = band.ReadAsArray(0,0,cols,rows)\

.astype(**float**).ravel()

G[:,b] = tmp - np.mean(tmp)

# матрица ковариации

C = np.mat(G).T\*np.mat(G)/(cols\*rows-1)

# Диагонализованная матрица

lams,U = np.linalg.eigh(C)

# Сортировка

idx = np.argsort(lams)[::-1]

lams = lams[idx]

U = U[:,idx]

# project

PCs = np.reshape(np.array(G\*U),(rows,cols,bands))

# Сохраняем результат

**if** outfile:

driver = gdal.GetDriverByName(’Geotiff’)

outDataset = driver.Create(outfile,

cols,rows,bands,GDT\_Float32)

projection = inDataset.GetProjection()

**if** projection is **not** None:

outDataset.SetProjection(projection)

for k **in** **range**(bands):

outBand = outDataset.GetRasterBand(k+1) outBand.WriteArray(PCs[:,:,k],0,0)

outBand.FlushCache()

outDataset = None

inDataset = None

# Здесь я ввожу именена сцен через консоль, но можно и сразу локально = “..”

infile = sys.argv[1]

outfile = sys.argv[2]

pca(infile,outfile)

Теперь для второй сцены все тоже самое

infile2 = sys.argv[1]

outfile2 = sys.argv[2]

dif\_image = abs(double(outfile-outfile2))

# h – число блоков пикселей в изображение(может быть разным) и задается юзером

vector\_s = zeros(rows\*cols, 3\*h\*h)

mean\_vector = mean(vector\_s, 1)

dif\_vector = vector\_s – mean\_vector

matrix\_cov = cov(dif\_vector)

eigvector = eig(matrix\_covar)

eigvalue = eigvalue(eigvector)

for k in range(length(eigvalue)

if(sum(eigvalue(k1:length(eigvalue)))>=

eigvector = eigvector(length(eigvalue))

feature = vector\_set \* eigvector

# K-means algorithm // немного псевдо кода

[label,~] = kmeans(feature,2)

CD = reshape(label, [row cols])

CD = CD – 1; // подгон размера

subplot(2,2,4);

imshow(CD,[])

title(‘CHANGE DETECTION’)

Заключение

Была проведена работа по анализу и выбору подходящей платформы для получения сцен. Были проанализированны различные подходы и методы реализации ОИ. Выбраны и реализованы два оптимальных алгоритма, которые наглядно демонстрируют возможности ОИ.

Перспективы данного проекта – огромны. Он может быть доработан во множестве аспектов, начиная с увелечения скорости получения и обработки сцен и заканчивая сложением всех результатов командной работы в приложение и создания собственной платформы ОИ.

Список источников

1. Antonio Rodriges. Страница описания проекта Change Detection. Режим доступа: <http://wiki.cs.hse.ru/Change_Detection> , свободный. (Дата обращения 1.01.2021)
2. ArcGIS. [learn.arcgis.com/en/arcgis-imagery-book](https://learn.arcgis.com/en/arcgis-imagery-book)
3. Masroor Hussain. Change detection from remotely sensed images: From pixel-based to object-based approaches / DongmeiChen, AngelaCheng, HuiWei, DavidStanley. (2013). [Электронный ресурс] / Режим доступа:  [https://www.researchgate.net/publication/258791369\_Change\_detection\_from\_remotely\_sensed\_images\_From\_pixel-based\_to\_object-based\_approaches](https://d.docs.live.net/b8c41b045aad9873/Документы/%20https:/www.researchgate.net/publication/258791369_Change_detection_from_remotely_sensed_images_From_pixel-based_to_object-based_approaches) , свободный. (Дата обращения 14.04.2021)
4. Jason Karl. Change detection / Anne Axel. (2013). [Электронный ресурс] / Режим доступа: <https://wiki.landscapetoolbox.org/doku.php/remote_sensing_methods:change_detection#:~:text=The%20goal%20of%20change%20detection,direction%20and%20magnitude%20of%20change> ,свободный. (Дата обращения 14.04.2021)
5. Chao Wang Hui Liu. High-Resolution Remote-Sensing Image-Change Detection Based on Morphological Attribute Profiles and Decision Fusion / Yi Shen, Kaiguang Zhao,Hongyan Xing and Haotian Wu. (2020). [Электронный ресурс] / Режим доступа: <https://doi.org/10.1155/2020/8360361>, свободный. (Дата обращения 14.04.2021)
6. Jin Chen. Land-Use/Land-Cover Change detection Using Improved Change Vector AnalysisPeng Gong. Chunyang He, Ruiliang Pu, and Peijun Shi. (2003). Режим доступа: <https://www.researchgate.net/publication/228907009_Land-UseLand-Cover_Change_Detection_Using_Improved_Change-Vector_Analysis> ,свободный. (Дата обращения 14.04.2021)
7. Earthexplorer.usgs.gov. [Электронный ресурс] / Режим доступа: [https://earthexplorer.usgs.gov/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fearthexplorer.usgs.gov%2F) ,через регистрацию. (Дата обращения 14.04.2021)
8. ENVI. [Электронный ресурс] / Режим доступа: <http://www.exelisvis.com/ProductsServices/ENVI.aspx> ,свободный. (Дата обращения 1.01.2021)
9. ERDAS. [Электронный ресурс] / Режим доступа: [http://www.erdas.com/Homepage.aspx](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fwww.erdas.com%2FHomepage.aspx) ,свободный. (Дата обращения 1.01.2021)
10. Feature Analyst. [Электронный ресурс] / Режим доступа: [http://www.vls-inc.com](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fwww.vls-inc.com),свободный. (Дата обращения 1.01.2021)
11. GDAL. [Электронный ресурс] / Режим доступа: <https://ru.wikipedia.org/wiki/GDAL> , свободный. (Дата обращения 1.01.2021)
12. Google Earth Engine. [Электронный ресурс] / Режим доступа: <https://code.earthengine.google.com/> через регистрацию. (Дата обращения 1.01.2021)
13. IDRISI Land Change Modeler. [Электронный ресурс] / [http://www.clarklabs.org/products/Land-Change-Modeler..](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fwww.clarklabs.org%2Fproducts%2FLand-Change-Modeler-Overview.cfm) ,свободный. (Дата обращения 1.01.2021)
14. OpenStreetMap. [Электронный ресурс] / Режим доступа: [https://www.openstreetmap.org/](https://www.openstreetmap.org/#map=3/69.62/-74.90) свободный. (Дата обращения 1.01.2021)
15. Landsat-8. [Электронный ресурс] / Режим доступа: <https://ru.wikipedia.org/wiki/Landsat-8> свободный. (Дата обращения 1.01.2021)

Рисунок 6. Результат работы алгоритма