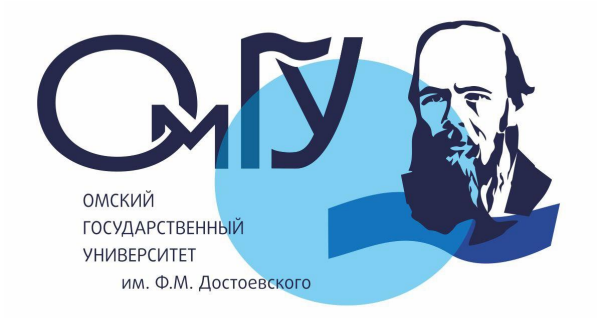
Использование python для решения прикладных задач дистанционного зондирования

Студент группы ФРМ-101

Никитин Максим Олегович



24 сентября 2022 г.

Цели и Задачи

Цель:

• Применение Языка программирования Python для решения задач дистанционного зондирования

Задачи:

• Анализ функционала языка программирования на предмет решения задач дистанционного зондирования

• Написание программного продукта, предполагаемого автоматическую загрузку блоков данных с открытых источников и последующий анализ этой информации используя SNAP.

Актуальность

Актуальность работы определяется тем, что технологии дистанционного зондирования Земли находят широкое применение при решении широкого спектра прикладных задач.

На данный момент международная спутниковая группа насчитывает сотни космических аппаратов преимущественно с оптическими сенсорами. Обработка информации предоставляемой нам спутниками очень важна, но затратна по времени. Для более быстрого и удобного анализа данных мы можем использовать блоки программного продукта, который рассмотрим в данной работе.

Python

Достоинтсва	Недорстатки
• Легко использовать и легко учиться. • Высокая портативность. Он может работать на любой платформе – на высокопроизводительных серверах и рабочих станциях. • Расширяемый язык с открытым исходным кодом. • Большой выбор встроенных библиотек. • Обеспечивает легкое взаимодействие с другими языками программирования.	• Это интерпретируемый язык • Python не подходит для разработки мобильных приложений. • Требует много памяти. • Есть ограничение на доступ к базе данных. • Он является языком с динамической типизацией, он показывает ошибку во время выполнения. • Незрелые торговые пакеты.

Области применения

Первые программные продукты, предназначенные для обработки данных дистанционного зондирования, распространялись исключительно на коммерческой основе.

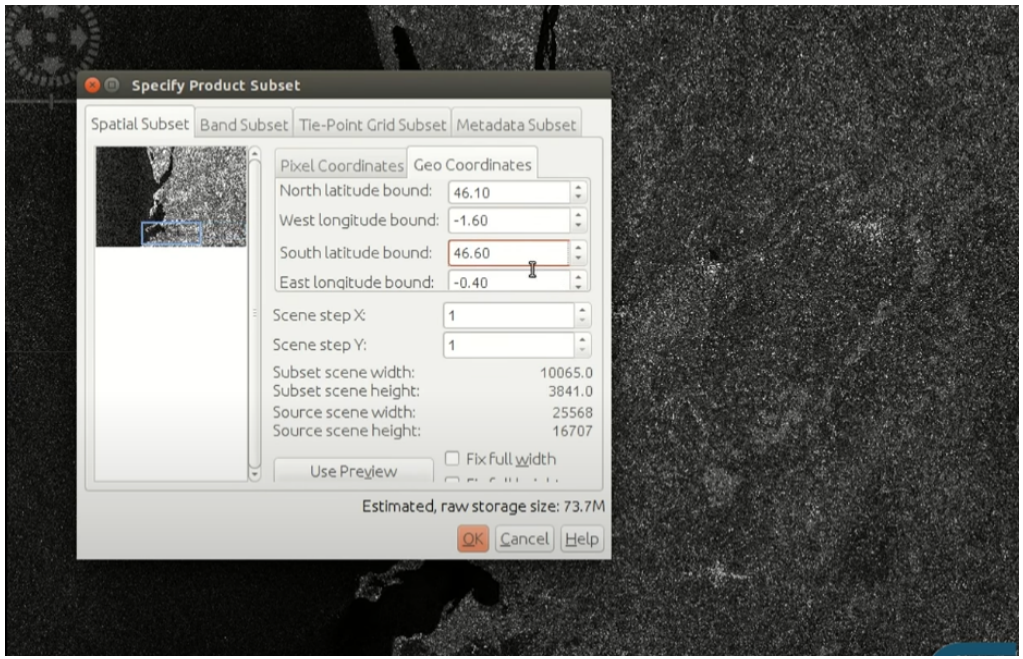
Однако в связи со сменой парадигмы предоставления данных дистанционного зондирования появились программные продукты, распространяемые космическими агентствами на бесплатной основе. Примером такого продукта является SentiNel Application Platform.

Блоки программного модуля

Была проделана работа по созданию программного кода, работающего на языке Pyhon. Полученный продукт предполагает автоматические загрузку блоков данных с открытых источников и последующий анализ этой информации используя SNAP.

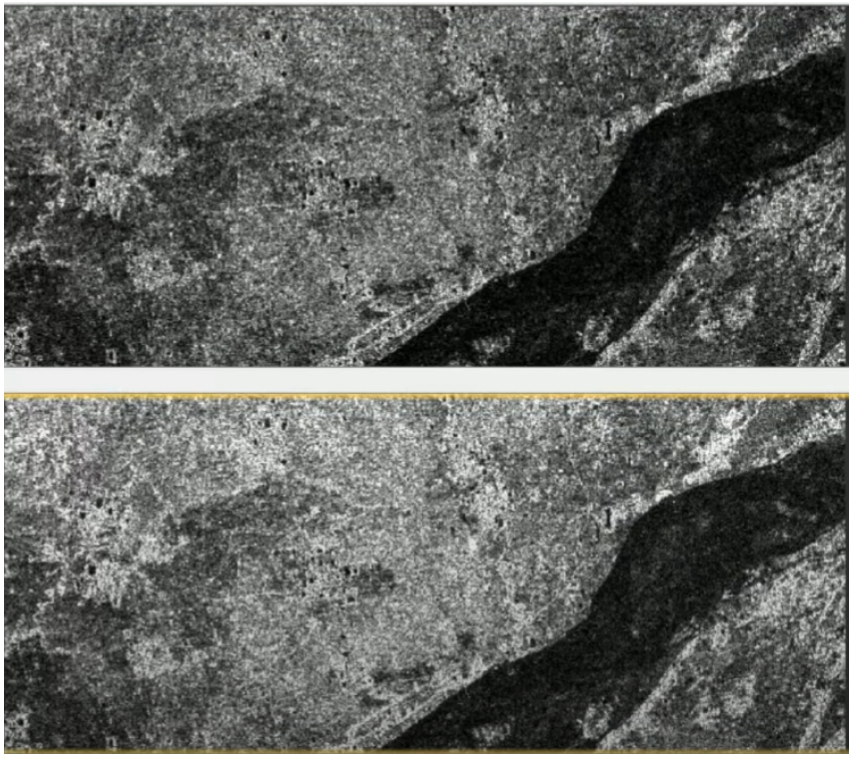
Было применено 6 этапов обработки сигнала:

- Подмножество (Subset)



Блоки программного модуля

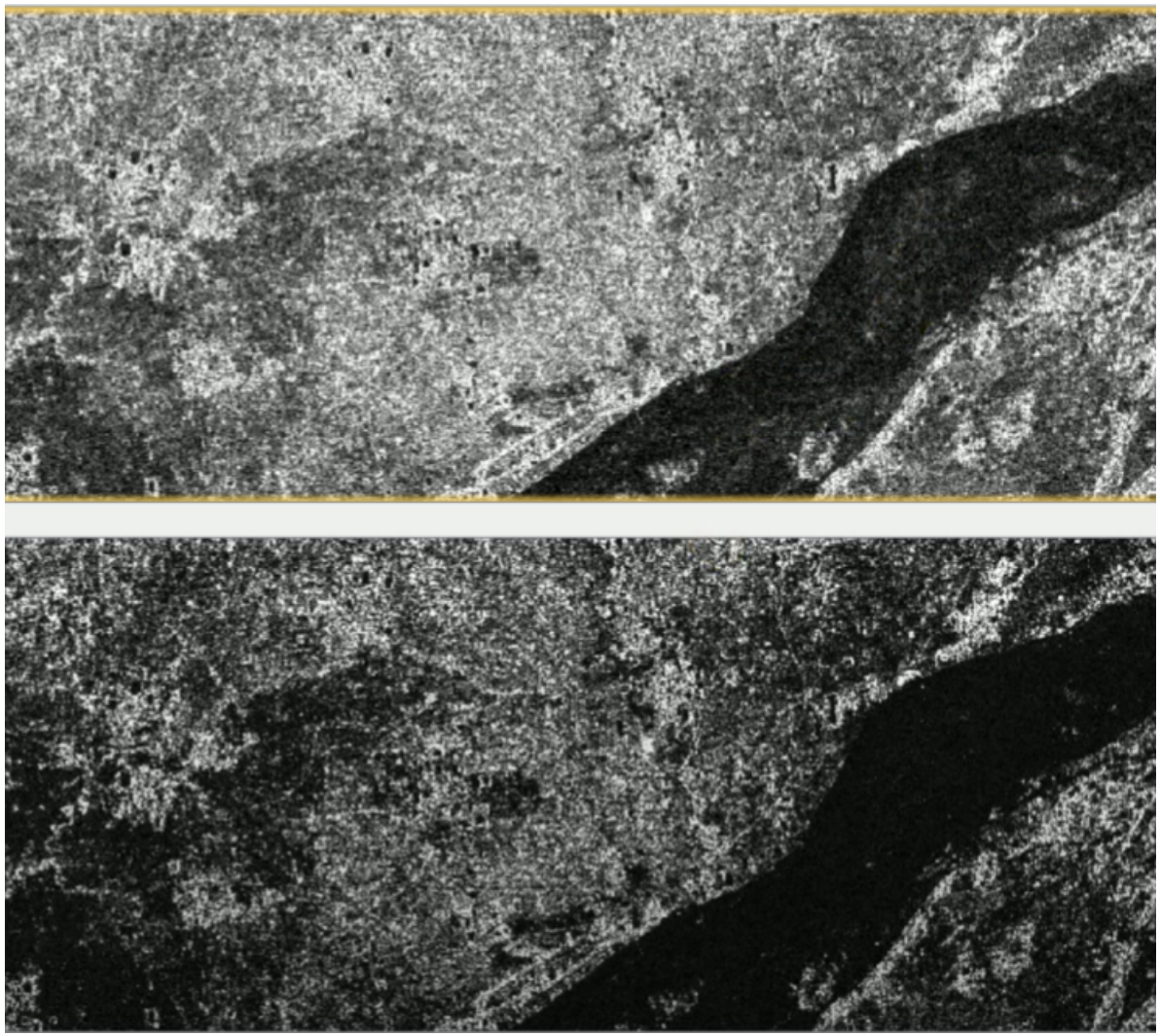
• Применение файла орбиты (Orbit File) - данные о точном расположении и скорости спутника.



• Фильтрация спеклов (Speckle Filtering) - это удаление пятен, которое избавляется от теплового шума, вносимого сенсором, с изображения, чтобы устранить потенциальные источники ошибок при анализе.

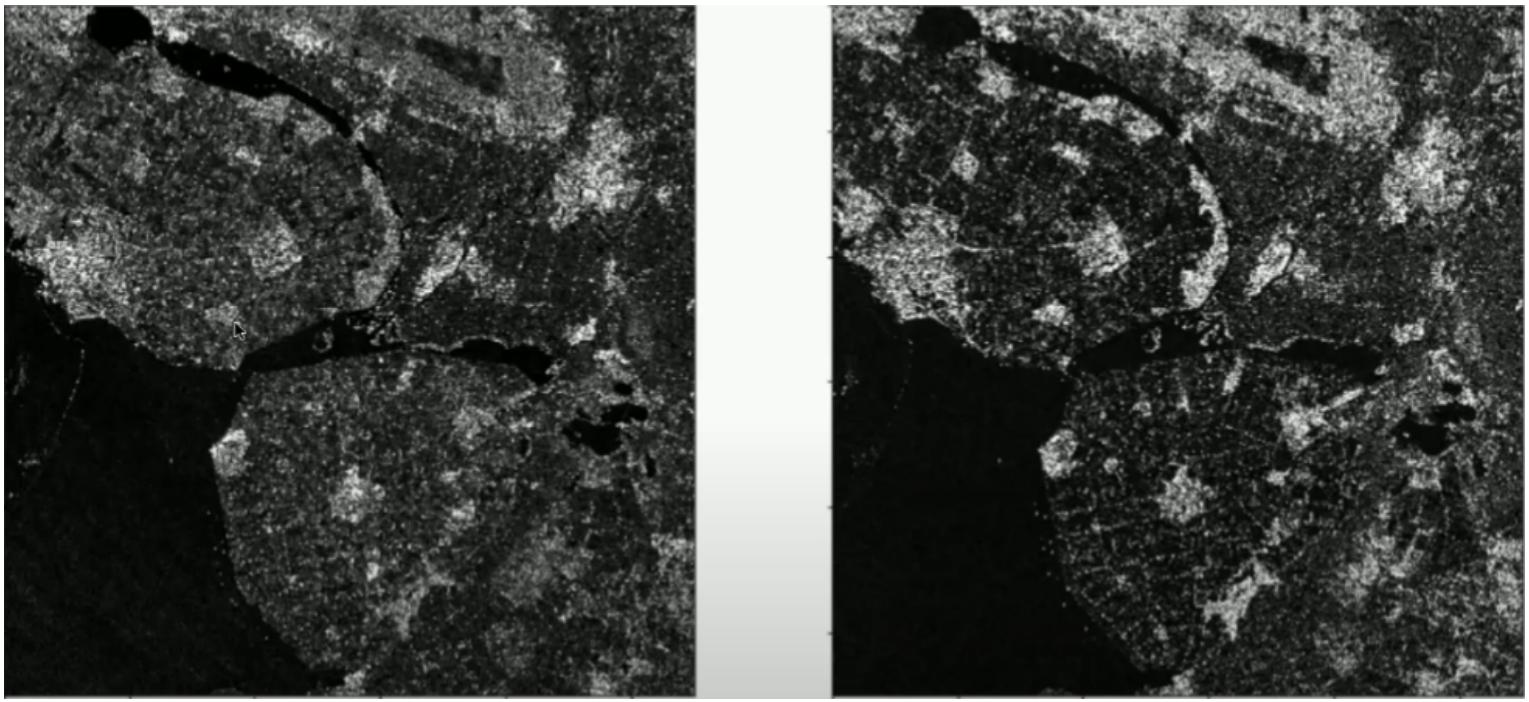
Блоки программного модуля

- Удаление теплового шума (Thermal Noise Removal)



Блоки программного модуля

• Радиометрическая калибровка (Radiometric Calibration). Этот шаг необходим для нормализации изображения, чтобы мы могли сравнивать несколько изображений во временном ряду.



• Коррекция рельефа (Terrain Correction) - является последним шагом для предварительной обработки изображения, он служит для того, чтобы убедиться, что все пиксели перемещены в правильные места.

Часть программного блока

ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Ф.М. Достоевского

```python
def add_subset(product):
    x, y, width, height = 12000, 8000, -1020, 2980

    parameters = snappy.HashMap()
    parameters.put('copyMetadata', True)
    parameters.put('region', "%s,%s,%s,%s" % (x, y, width, height))
    subset = snappy.GPF.createProduct('Subset', parameters, product)
    # list(subset.hetBandNames())
    return subset;

def apply_orbit_file(product):
    parameters = snappy.HashMap()
    parameters.put('Apply-Orbit-File', True)
    apply_orbit = snappy.GPF.createProduct('Apply-Orbit-File', parameters, product)
    return apply_orbit

def thermal_noise_removal(product):
    parameters = snappy.HashMap()
    parameters.put('removeThermalNoise', True)
    thermal_noise = snappy.GPF.createProduct('ThermalNoiseRemoval', parameters, product)
    return thermal_noise

def radiometric_calibration(product):
    parameters = snappy.HashMap()
    parameters.put('outputSigmaBand', True)
    parameters.put('sourceBands', 'Intensity_VV','Intensity_VH')
    parameters.put('selectedPolarisations', 'VH','VV')
    parameters.put('outputImageScaleInDb', False)
    calibrated = snappy.GPF.createProduct('Calibration', parameters, product)
    return calibrated

def speckle_filtering(product):
    parameters = snappy.HashMap()
    parameters.put('filter', 'Lee')
    parameters.put('filterSizeX', 5)
    parameters.put('filterSizeY', 5)
    speckle = snappy.GPF.createProduct('Speckle-Filter', parameters, product)
    return speckle
```

Выводы:

- В работе был произведен анализ функционала языка программирования на предмет решения задач дистанционного зондирования
- Был написан программный продукт на языке Python, который расширяет и автоматизирует процесс предварительной обработки спутникового снимка Sentinel-1
- Данный программный модуль можно внедрить в состав технологического ПО, тем самым значительно сократить время работ, повысить точность, а также облегчить труд специалиста.

Спасибо За Внимание