West University of Timisoara

Faculty of Mathematics and Computer Science

Study Program:
Computer Science in English

Bachelor Thesis

COORDINATOR:

Conf. Dr. Marc Eduard Frîncu

GRADUATE:

Bulz Gabriel

Timişoara
2018

West University of Timisoara

Faculty of Mathematics and Computer Science

Study Program:
Computer Science in English

Prediction of areas with high flooding risk using physical models

COORDINATOR:

Conf. Dr. Marc Eduard Frîncu

GRADUATE:

Bulz Gabriel

Timişoara
2018

Abstract

Floods are without doubt the most devastating natural disasters, striking numerous regions in the world each year. During the last decades due the increased frequency of heavy rain and a continuously increasing concentration of population near water regions a lot of assets and lives were lost.

That is the reason why a system that can create a flood forecasting is

To process the resulted images we will use a library in python called GDAL which can handle that specific type of files (.tif)

In the first part we will try to identify the areas nearby the rivers and lakes (because that areas are more likely to be flooded), and in the second part we will try to analyze the images based on the height of each section. The water from the rain will gather in the lower areas, so we can presume that that areas are likely to hold water.

Chapter 1

Introduction

1.1 Motivation

Water is an essential component of ecosystems for the sustainability of life on our planet. It balances ecosystems and maintains climate variation, carbon cycling, etc. It is equally important to humans and other forms of life. Its excess or absence could lead to disasters and extreme land use change. Hence, identification of water bodies is an essential process in science and engineering research. The identification can be useful in various ways, such as estimation of water areas and demarcation of flooded regions [1, 2].

Floods are one of the most devastating natural disasters, striking large regions in the world each year. During the last years due the increased frequency of heavy rain a lot of assets and lives were lost.In general, less developed countries shown the most vulnerability to floods, causing damages that significantly affect the national GDP. At country and community levels important initiatives have and are being devoted to implement appropriate countermeasures, both structural and non-structural, aiming to alleviate the persistent threats of water-related disasters. [3]

Flood prediction models are of significant importance for hazard assessment and extreme event management.Robust and accurate prediction contribute highly to water recourse management strategies, analysis, and further evacuation modeling [4].

Thus, the importance of advanced systems for short-term and long-term prediction for flood and other hydrological events is strongly emphasized to alleviate damage [5]. However, the prediction of flood lead time and occurrence location is fundamentally complex due to the dynamic nature of climate condition. Therefore, today's major flood

prediction models are mainly data-specific and involve various simplified assumptions [6].

Physically based models were long used to predict hydrological events, such as storm, rainfall, water flow models, and other global circulation phenomena , including the coupled effects of atmosphere, ocean, and floods. This is the reason why we chose to develop our application based on a physically flood prediction model. Other types of prediction models are data-driven models e.g. machine learning or hybrid models which can combine data-driven, statistical and physically base models.

Physical models showed great capabilities for predicting a diverse range of flooding scenarios [7],especially in long and mid-term prediction, although they often require various types of geomorphological and hydrological data.

In contrast to the physically based models, the data-driven prediction models using ML shown a higher performance rate on short-term forecasting compared to the physically based models. In addition, it was shown that the performance of ML could be improved through hybridization with other ML methods, soft computing techniques, numerical simulations, and physical models. Such applications provided more robust and efficient models that can effectively learn complex flood systems in an adaptive manner. Although the literature includes numerous evaluation performance analyses of individual ML models [8, 9, 10, 11], there is no definite conclusion reported with regards to which models function better in certain applications. In fact, the literature includes only a limited number of surveys on specific ML methods in specific hydrology fields [12, 13, 14]. Consequently, there is a research gap for a comprehensive literature review in the general applications of ML in all flood resource variables from the perspective of ML modeling and data-driven prediction systems.

Although the data-driven models using ML can be much efficient in some cases there is a still a big draw back regarding their development and use. For a data-driven model to have a high accuracy it will be needed a very large amount of training data, which can be hard to acquire due the weather conditions and monitoring devices availability time. Furthermore the development of a data-driven model using ML is very expensive because it requires a complex model which needs to be trained for a long period of time, requiring a high computation cost, and it also requires a longer validation, testing, and evaluation period.

Even dough the data-driven models using ML can be a great scientific tool they tend to be hard to understood by a non trained person, and they can require, as shown

above, a high run cost and a data set which can be very difficult to acquire by every person. This is why we chose to develop a physically prediction model which can be used by everybody with a low run cost, which requires a minimum data-set easy to obtain via different services like Sentinel 1,2,3 or Landsat satellite images programs. This app has the potential to serve regular people when making decisions about where to buy a property, or where to build a house without any risk saving lots of money.

1.2 Our Contribution

1.2.1 Method and Outline

This paper identifies the state of the art of physically methods for flood prediction taking into account the processing time, cost, efficiency and difficulty to use. The methods that we used shown a very significant performance and accuracy rate.

The applications in flood prediction can be classified according to flood resource variables, i.e. river flow, flood peak discharge, urban flood and plain flood. Among these key influencing flood resource variables, and the spatial examination of the topographical images, the application include the possibility to detect with high accuracy the water surfaces (over 89% accuracy from all tests).

The mainly methods that we used in the application were the detection of water and flooding a land area based on the topographical images. The water detection was obtained by combining a set of near infrared (or NIR) and green band satellite images, and then applying a method of extraction called normalized difference water index (NDWI). This method is based on the extraction the water bodies taking into account the reflectance property of water.

After the water has been detected, the land area is flooded based on the topographical map of the surface. One of the problems that we encountered here is ,besides the water detection, the fact that the topographical satellite images tend to cover a bigger area than the NIR and green band area from the satellite images, so we had to map the smaller NIR images into the bigger topographical images, but we will discuss this problem and how we solve it in the following chapters. The resulting product of our application is a area that has a high possibility of being flooded during a heavy rainstorm, results that can be achieved using a very low processing time, cost and resources.

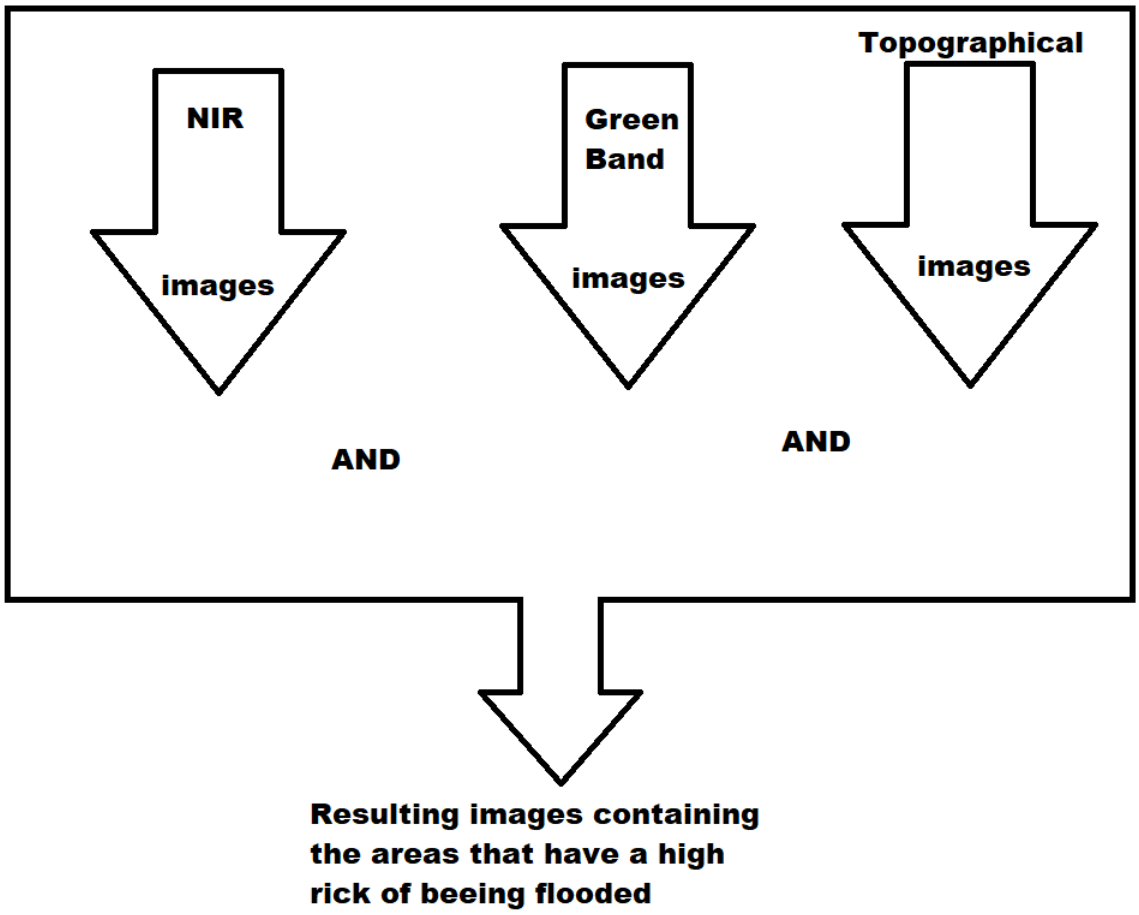


Fig 1

Combining those techniques with an easy to use and understand client interface we managed to create a system that can predict the land areas with a high potential of being flooded. The processing time and cost are reduced because the server part has to go through a set of images only twice, first time to detect the surface water and second time to flood the land area based on the topographic map, so the computations are reduced as low as possible. On the client side, there are big advantages because the set of resources (satellite images) are free and easy to access and the results are easy to understand, making this application ideal for a non trained person.

1.2.2 State of Art of Physically models in flood prediction

For creating our physically prediction model we chose to use NDWI water extraction technique due its ease use and low processing time. McFeeters [15] developed the normalized difference water index (NDWI) using the reflectance of the green (band 2) and near-infrared (band 4) bands of Landsat TM (Thematic Mapper). NDWI is one of the most widely used water indices for a variety of applications, including surface water mapping, land cover analyses [16, 17, 18] and also it shown great classification accuracy in areas that include shadow and dark surfaces.

Besides the existence of NDWI index for detecting the water bodies, there are other techniques presented in various papers [19, 20] that shown significant results. These methods are the Multifractal analysis of water bodies using Wavelet method and Fluctuation analysis and the detection of water using neural networks. Both of these techniques shown a high accuracy (multifractal analysis shown the best of those two, over 89%; very similar to the accuracy obtained by our NDWI index), but there are some downsides regarding the neural network water detection. That method shown great results but only on some particular sets of images and it did not provided a proper classification for all radar images. The studies [20] suggests that the neural network errors may appear due to the high sensitivity to the noise in the images.

On short, after our model manages to detect the water areas from a picture it starts to expand these areas taking into account the topography of the land, like a flood bases in the land level. This model is called a physically prediction model and it allows us to create a simply to understand and test model, which will be time and cost efficient. The image has to be processed only two times in a linear manner, first time to detect the water areas with a complexity of O(n) = N , and second time to flood the land area with a complexity that can go up to O(n) = N^2.

There exists more complex physical models that can take into account more elements, like land water saturation, vegetation type of that area and the volume of rain that will fall over specific areas, but that types of models are more complex, and more expensive both on execution time and on development time. The test and validation part will need to be more complex, and the input required will be harder to gathered up by a simple user.

Besides the physically prediction models there are some other forecasting models presented in various papers [21], like data driven models based on machine learning and

hybrid models that combine physically, and machine learning hydrologically models.

The first one that we will discuss is the **ML-based hydrology-dynamic modeling**: The computational cost of this model can limit the resolution and the scope of the hydro-dynamic system. The costs can be reduced by up to orders of magnitude by taking advantage of machine learning derivation methods, as it was experimentally done of other fluid dynamics models [22].

The second model that we will discuss about will be the **Remote discharge estimation**: The most important obstacle for incorporating machine learning into the hydrology field is the limited data. Measuring the water levels and the discharges is relatively difficult when it comes into attention the global order of 100,000. But still, the quantity, variety and quality of satellites constantly imaging all river is rising at a very fast pace. The ability to estimate the river discharges without in-site measurements is a a task that has become both critical for the hydrology field, and incredibly well suited for machine learning field.

The third and the last model that we will discuss about is the **Hybrid physics-machine learning hydrologic models**: This model is a alternative to the machine learning approach, and it adopts a hybrid approach where a ML model is combined with the classical physics-based components, which can enclose important prior knowledge about the domain structure. Some could view this approach as allowing the physically model to capture the major hydrologic model, while the ML will be responsible for error correction, and calibration of the final model. As an interesting fact, it was shown that the physically models tend to be more accurate on the long-term and mid-term prediction, while the ML tend to be more accurate on the short-term predictions, and by combining those two techniques in one we can possibly obtain a accurate prediction model on both long-term and short-time periods.

1.3 Thesis Structure

Chapter 2

The application

2.1 Technologies used

2.1.1 Collecting and processing the resources

Collecting the resources

Our prediction model was created with a simple scope in mind, to offer a simply application that can predict the areas with high risk of being flooded during a rainstorm, with as little processing time and cost as possible. The input for our model had to be easy to obtained and the output had to be easy to understand, so we chose to use a input data that is both easy to obtained and free (it will need to be accessible by every person). The input for our model is a set of .tiff images that should contain a near infrared image (NIR), a green band image, and a topographic image. A simple way to obtain the NIR and green band images is by using the available data offered by satellites like Landsat 1 to Landsat 8 or Sentinel 2.

The Landsat project is one of the longest-running enterprise projects for acquisition of satellite imagery of Earth. The project was first developed by NASA and then it was transferred to NOAA (National Oceanic and Atmospheric Administration) by U.S Jimmy Carter's presidential directive. There were 8 Landsat satellites on the Earth's orbit, now only 2 of them are still active, and the resources collected are available on the USGS (U.S Geological Survey) website. The only thing that a user will need in order to access the data is a free account. After the account is created the user can select different areas via USGS's EarthExplorer portal and download the area's satellite imagery using filters like

Data Range or satellite preference (Landsat satellite 1-8).

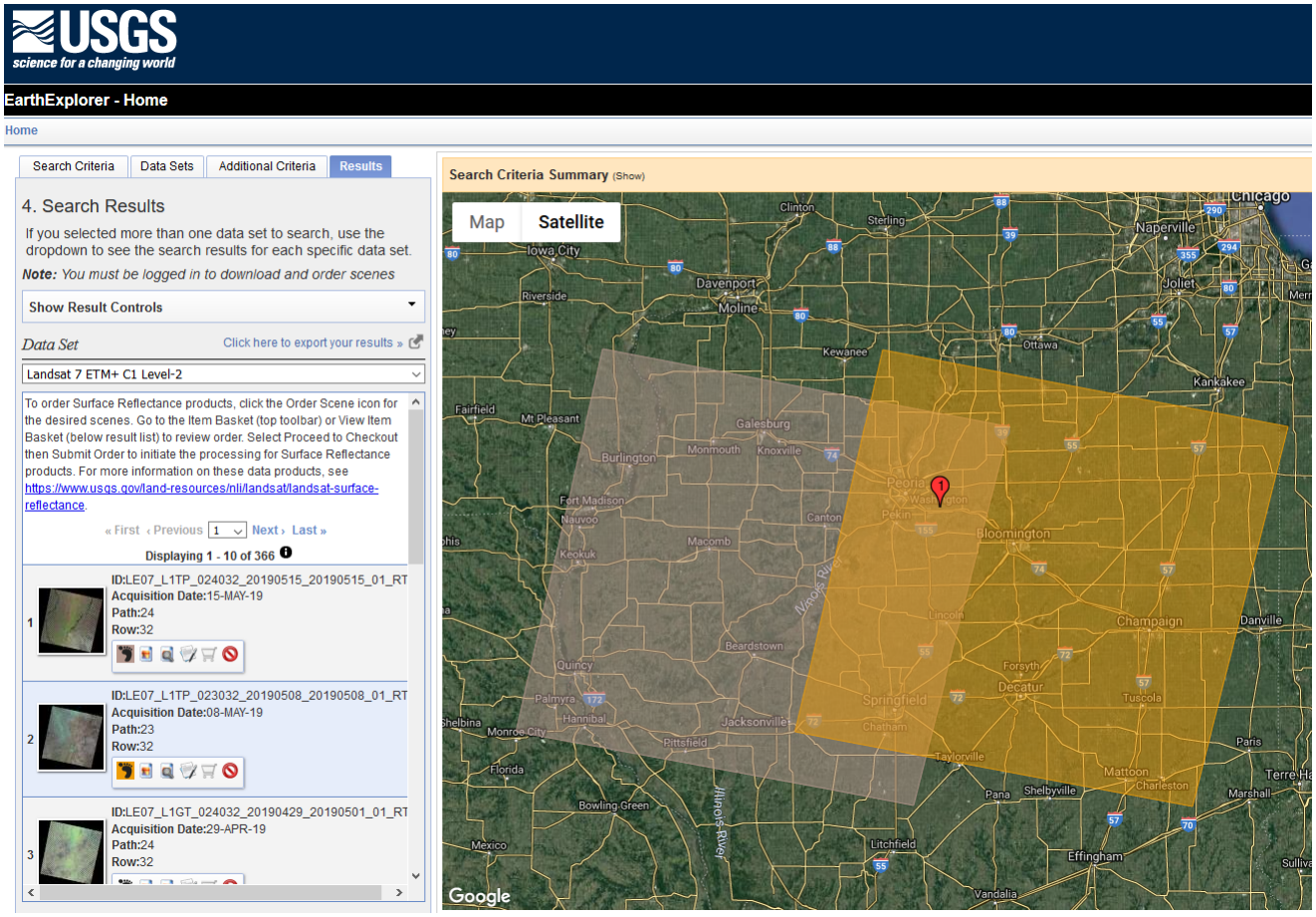


Fig 2 - [23]

The image presented above (fig 2) is a shot taken while searching for our test data from a region in U.S (Cairo) that was heavy flooded during a rainstorm by the Mississippi river. The yellow and pink areas represents two different scenes captured by the Landsat 7 satellite. A set of 8 images are found in a package downloaded from this website depending on the satellite that had provided them, but all the Landsat satellites imagery set contains a NIR and green band shot of the selected area (the main difference between them is the resolution in meters of the photo). We used in our research mainly Landsat's 7 images because the photos are at a higher resolution (30 m) and most of them are already processed by the USGS servers (some images need to be processed by the internal servers before being available to the users).

Landsat 7 Enhanced Thematic Mapper Plus (ETM+)

Bands	Wavelength (micrometers)	Resolution (meters)	Bands	Wavelength (micrometers)	Resolution (meters)
Band 1 - Blue	0.45 - 0.52	30	Band 5 - SWIR 1	1.55 - 1.75	30
Band 2 - Green	0.52 - 0.60	30	Band 6 - Thermal	10.40 - 12.50	60* (30)
Band 3 - Red	0.63 - 0.69	30	Band 7 - SWIR 2	2.09 - 2.35	30
Band 4 - NIR	0.77 - 0.90	30	Band 8 - Panchromatic	0.52 - 0.90	15

Fig 3 - [24]

As we specified above there is another option to obtain the images besides Landsat, and that is through the Sentinel program. Sentinel is an observation mission from the Eu Copernicus Programme that acquires optical imagery of Earth at large resolution (10 to 60 m) over land and coastal waters. There are 3 satellites launched by the Copernicus Programme but we will focus mainly on Sentinel 2, because it has the largest collected data.

The Sentinel 2 satellite has a multi-spectral instrument with 13 spectral channels in the visible, short wave and near infrared spectral range. The bands that are interesting for us are the 3-rd band which is the green one and the 8-th band which is the NIR one (the bands have a spatial resolution of 10 m). The images can be downloaded from the Copernicus Scihub website, again by creating a free account. There is a possibility to select a date range, cloud coverage in percents and the satellite platform which is the best suitable for the user needs (I.E Sentinel 1,2 or 3).

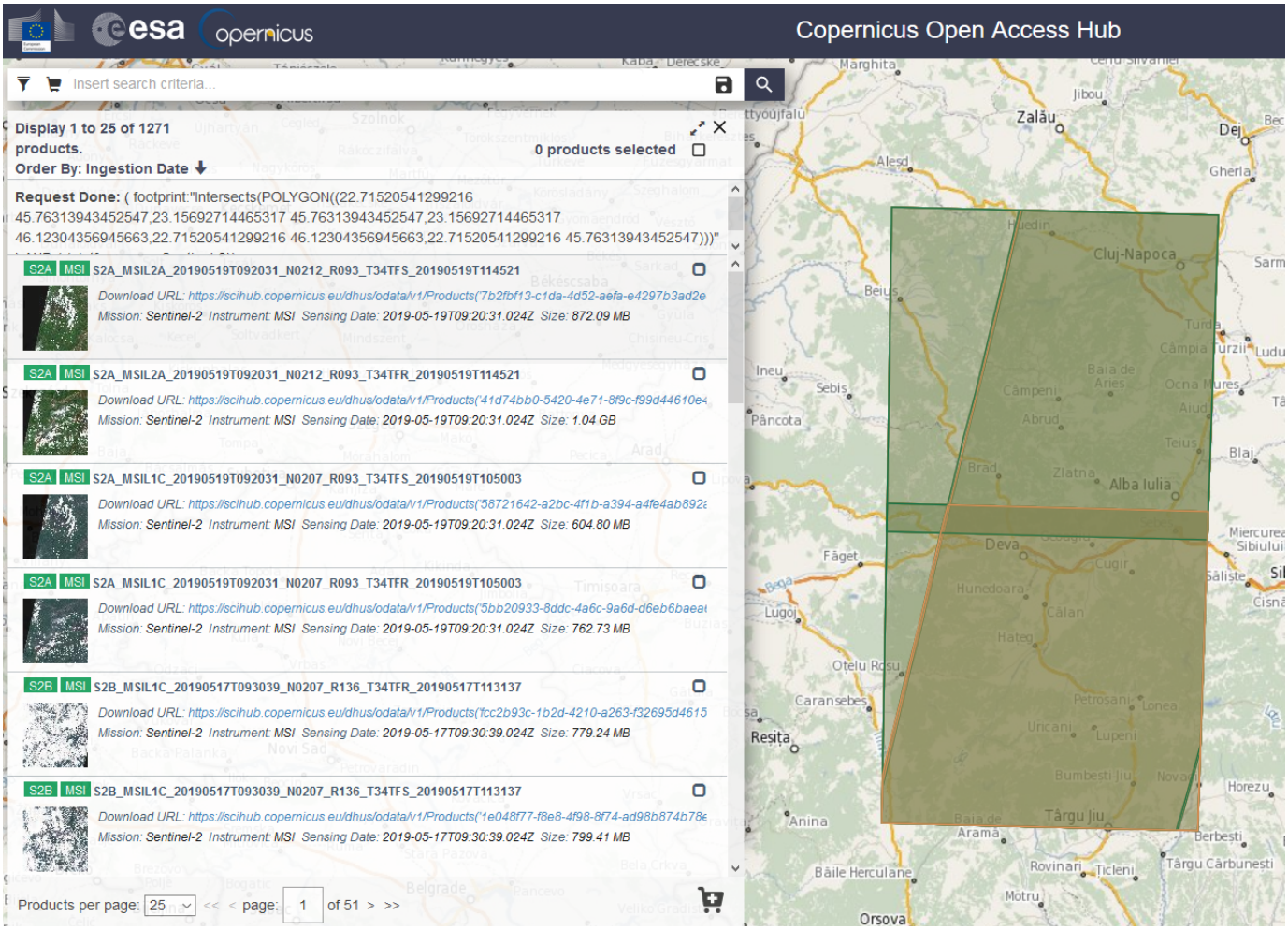


Fig 4 - [25]

Still, there are some drawbacks from using the Sentinel imagery. We chose to use the Landsat images over Sentinel's data set, because the Sentinel 1 images are at a lower resolution, and the Sentinel 2 and 3 images are too recent taken (the satellites were launched in 2015, so the data could not cover a larger time period for our data analysis).

After we obtained the NIR and green band images we will need to download a topographic map of the area ,that will be processed. The topographic map can be taken from USGS website by searching for digital elevation maps. The process is pretty similar to the one of collecting Landsat imagery. A interesting difference between a NIR and a topographic image is represented by the images size. The topographic image is usually taken over a larger field, and this is a problem that we encountered during the processing of the areas with high flood risk. We needed to determine where to place the smaller image inside the bigger one based on coordinates, but we will discuss about this problem in one of the following sections.

Processing the resources

Before a set of images can be processed by our application they need to be rendered. For this process we used a program called QGIS. Qgis is an open source geographic information system, created by OSGeo (Open Source Geospatial Foundation), licensed under GNU, that supports numerous vector, raster and database formats and functionalities [26].

The images will need to be rendered from 8 bit depth to 32 bit depth and the pixel range value will be mapped between 0 and 255. They will also need to be saved with .TIF extension, to be later processed by our servers.

The Sentinel images are pretty straight forward to rendered, but the images from Landsat will need an extra step before being ready to save. The Landsat 7 images have what seems to be "black stripes" across the side of the image. This is due to a failure of Landsat 7 Scan Line Corrector. The forward movement of the satellite on the orbit should be compensated by The Scan Line Corrector, but because of the failure a zigzag pattern of ground tracking is used instead of a mapping in straight lines. This can be seen very clearly in the image below (fig 5), and the location of the black stripes varies between 390-450 m; therefore US Geological Survey(USGS) estimates that affected images lose about 22% of their data [27].

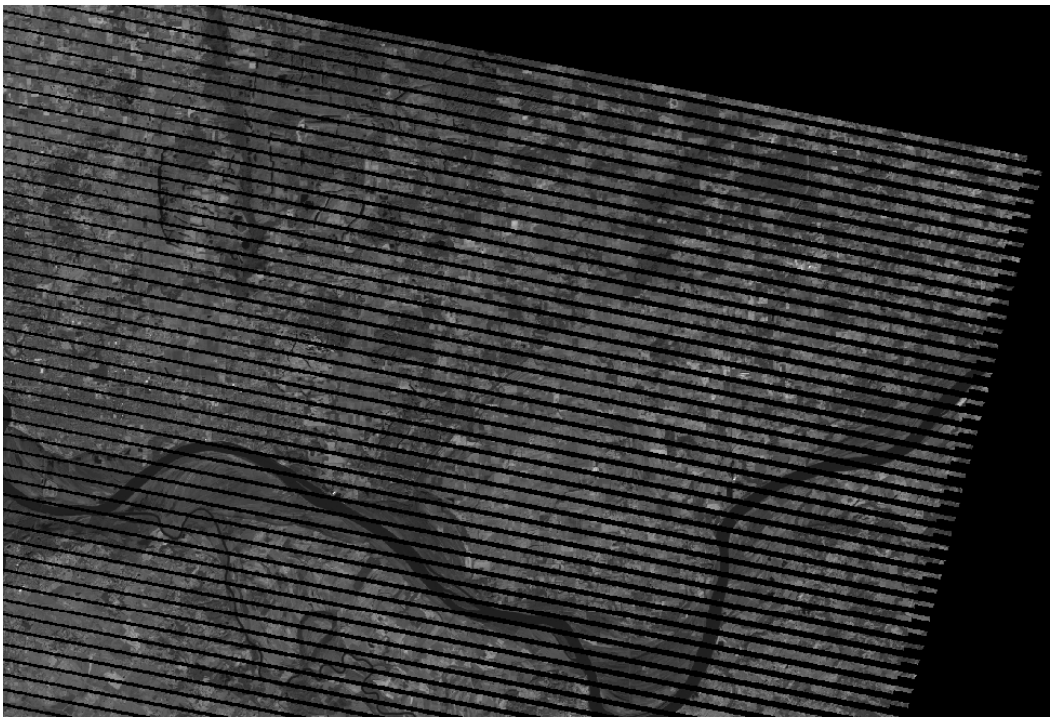


Fig 5

The black lines get smaller as we approach the center of the image, which means that we can crop some parts of the land area and use them without any future modification, or if we need to use the full image we can try to apply a image correction, which should fill the black lines based on the gap masks offered by Landsat (for this part we will need to have Gdal installed). Below we can see how a mask layer looks like (fig 6).

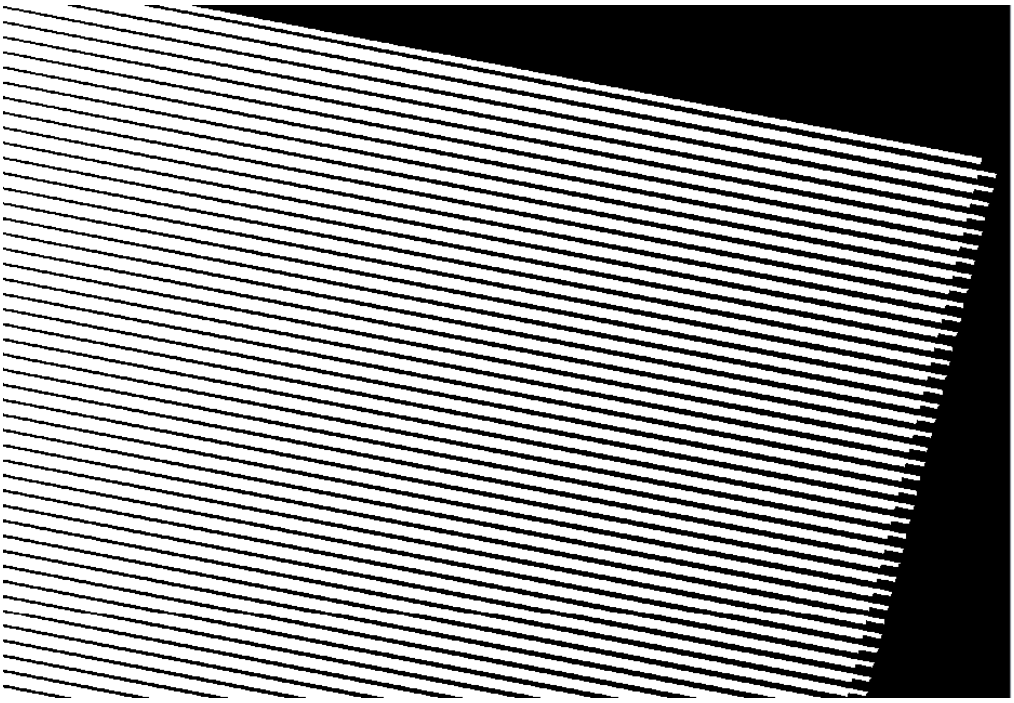


Fig 6

After the images are ready they can be rendered by saving the image with .tif extension and checking the "render image" box.

2.2 Programming languages and Frameworks

2.2.1 Programming language: Java

Java is popular object-oriented programming language that was designed to have as few implementation dependencies as possible. It was created in 1995 by Sun Microsystems and now it is owned by Oracle.

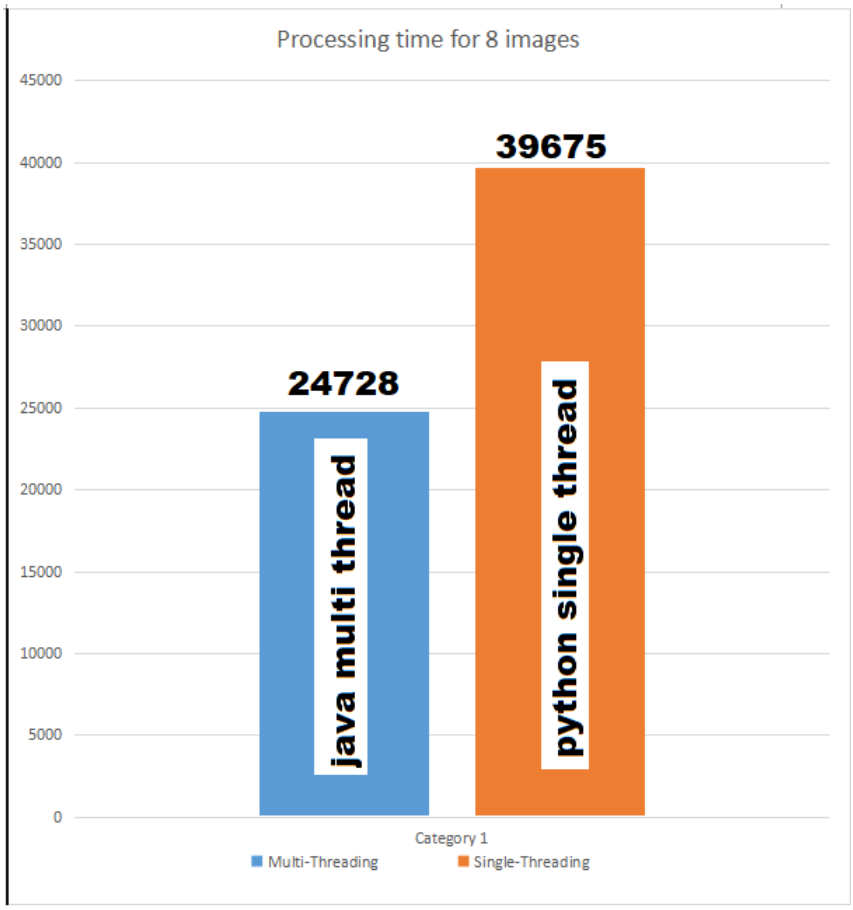
This project was developed using Java 1.8 with two external libraries i.e.: Gdal and jai-imageio-core 1.4.0. Any version below 1.6 will now work as intended because of jai-imageio-core library. These libraries were used to process the TIFF images from satellite.

At first we intended to use Python as the heavy lifting programming language because of the more relaxed syntactic structure, but in the end we chose Java because of its parallel programming capabilities. Usually when we have to work with satellite images we should take into account the fact that the images can be really large, like 8161x7211 pixels in size covering over 589.000.000 km^2, and when we have to process more that one image of this kind the processing time becomes very important.

Python will become much slower in this area because of the Global Interpreter Lock or GIL (a mutex, or a lock, that allows only one thread to be in a state of execution at any point in time). The impact of the GIL is visible only to developers who execute multi-threaded programs, because it can create a performance bottleneck at the CPU level.

In the following image (fig 7) we can take a look over the processing time between the Java's multi-threaded system and the single-threaded Python's system

Fig 7



The tests were made by us on a sample of 8 images with an average size of 1000x1000 px, and the results shown that the java multi-threaded system (marked with blue on fig 7) was in average faster with 40% than the python single-threaded system. We valued this feature very much because the reduced time of processing combined with the image handling power offered by java helped us to create a time and cost efficient application, and that is what we intended at first.

2.2.2 Programming language: Python

Python is a interpreted, high-level, open source programming language that was made to have a relaxed syntactic structure and to be more easy to read. It supports both functional and object oriented programming and its mainly targeted to a fast development and easy to maintain code.

Python had played a major role in the project because of its relaxed syntactic structure and it was use to create the server part of the application, using Flask (flask is a microframework for python web based applications).

After the server receives a set of satellite images from the user, a Java process is started by the server to solve the request. The java files are complied when the server is started for the first time and then for every request a java process is started. All this part was handled using the python's "subprocess" library

We will take a short look on how the python server compiles the java files and how a process is started, and we will discuss in more details in the next chapters.

Here we can take a short look on how the java files are compiled

```
import subprocess

def compile_java_files():
    """
        compile all java files that are used for image processing
        and link all libs
    """
    java_files = JAVA_FILES_PATH + '\\*java'
    java_libs = JAVA_LIBS_PATH + '\\*'
    subprocess.Popen(['javac', '-d', JAVA_OUT_COMPILED_CLASSES,
                      '-sourcepath', JAVA_SOURCEPATH_CLASSES,
                      '-cp', java_libs, java_files], shell=True, stdout=True)
```

Fig 8

Here we can take a look on how a process is started

```
def process_files(path_files, result_directory):
    """
        run a java process that will solve the request
        param : path_files - path to the folder where images have been unziped
        param : result_directory
        return : 1 - if the process have failed
                 0 - if the process had succed
    """
    java_libs = JAVA_LIBS_PATH + '\\*'
    compiled_libs_and_classes = JAVA_OUT_COMPILED_CLASSES + ';' + java_libs
    java_main_class = JAVA_PACKAGE_NAME + '.' + JAVA_MAIN_CLASS
    cmd = ['java', '-cp', compiled_libs_and_classes,
           java_main_class, path_files, result_directory]
    stdin = PIPE
    stdout = PIPE
    stderr = STDOUT
    proc = subprocess.Popen(cmd, stdin=PIPE, stdout=PIPE, stderr=STDOUT, shell=True)
    stdout, stderr = proc.communicate()
    output = str(stdout).replace('\\r','').split('\\n')
    final_output = str(output[0]) + str(output[1]) + str(output[2])
    correct_output = JAVA_OUTPUT_DETECT_CLASS + JAVA_OUTPUT_PREDICT_CLASS + JAVA_OPTIONS

    if (final_output == correct_output):
        return 0
    return 1
```

Fig 9

2.2.3 Gdal Library

Gdal (Geospatial Data Abstraction Library) is a software library available for multiple programming languages (python, java, c++) for manipulating rasters and geospatial data vectors. It was released by the Open Source Geospatial Foundation (OSGeo) in the 2000 year with a permissive X/MIT free software license[28].

We used Gdal as a Java library to process the satellite images. We needed a way to extract the coordinates of specific pixels from .Tif images, and The Gdal library offered a convenient way of doing that by using gdal.Datasets, gdal.Band and the GeoTransform matrix.

2.2.4 Flask Framework

Flask is a web framework written for Python which is classified as a microframework because it does not require particular tools or libraries, and it is really light weight because it has no database abstraction, no user authentication and no form validation.

Flask was developed based on the template engines offered by Jinja 2, which can enable a accelerated development for our web application. It also offers a micro but extensible web framework, which can become very flexible by using several widely used web development tools and libraries [29].

There is an another available web framework dedicated to Python which we took into consideration when we built this application and that is Django. At first look Django seemed a better choice than Flask, because it has a better template engine than Flask, it has a ORM (object-relational mapping) system that can work with several databases like SQLite, Oracle and MySQL and it has a built-in bootstrapping tool, but despite all this upsides, which can make the application development much easier, there are some drawbacks when using Django. Even though Django could offer a cheaper App development, we should also take a look on how a Django HTTP server will compare to a HTTP Flask server.

We had to consider the fact that the machine on which our application will run will need to be cheap to maintained, and it should be pretty responsive, which means that it should have a low latency (that was one of our main focuses). We took a look on several benchmarks that compared the performance obtained by Flask and Dajngo during longer sessions of run time, because the available resources tend to change with time. We had to determine how they can handle a more intensive environment where the available resources can decrease. Like on the real machines where the decrease of resource is non linear because the OS can spawn daemons or trigger some side jobs, we had to find some tests that will take this facts into account, and we found that the HTTP Flask server performs much better than the HTTP Django server in this king of conditions. The tests that we take into consideration [30] were made using Gunicorn (a Python Web Server Gateway HTTP interface) at a constant 4000 queries per seconds. In the tests the HTTP Falsk server displayed a lower latency and error numbers and a higher request number compared to the HTTP Django server.

	Requests/s	Errors	Avg Latency (s)
Django+Gunicorn	1799	26883	97
Flask+Gunicorn	2714	26742	52

Fig 10 - [30]

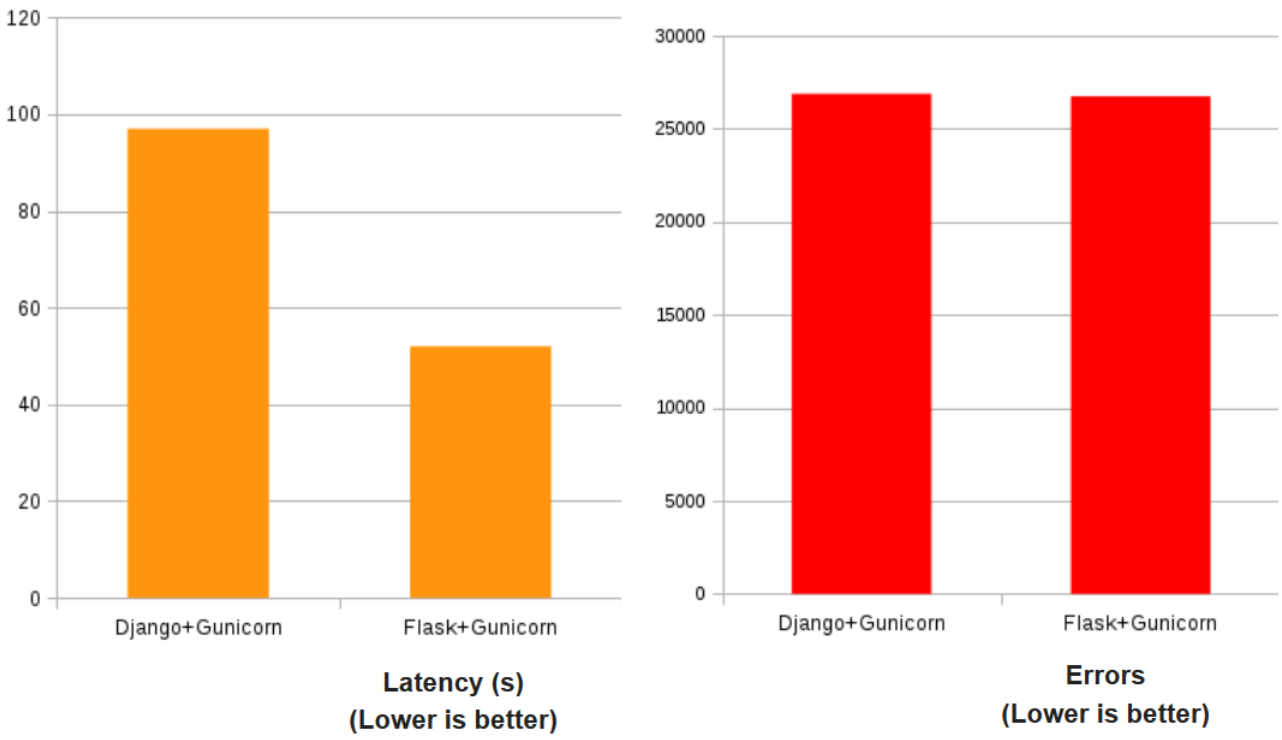


Fig 11 - [30]

2.3 Functional description

In this section we will take a closer look on how our application manages to process the input offered by the user and predict the areas with high flooding risk. Firstly we offer a friendly and easy to understand web interface where any user can upload a set of satellite images, where a set is composed of a NIR image, a green band image and a topographical image (a user can upload multiple sets of this kind at once; the sets will be identified based on names).

After the input is uploaded to the server a solving request will be created and placed on a queue. Each request is processed one at a time, and when it turn will come, the server will unzip the input and start the prediction. The first step in our prediction consists in detecting the water surfaces by combining the NIR band image and the green band image and then applying a formula called NDWI (Normalized Difference Water Index) to each pixel. After that, the resulting image is mapped to the topographical image and a flood algorithm is applied. The flood will start from the water bodies already detected and it will spread base on the land's topography.

2.3.1 Detecting surface water using NDWI + referinte si procente

The detection of the water bodies was probably one of the most challenging part of this application, because there are different types of water, with different colors and different densities. At first, we tried to detect the water surfaces based on their color, but usually the chromatic of the oceanic water is very different than the chromatic of the lakes and rivers. The color range that we had to cover was too large, so we gave up on this idea.

The second method that we tried was to use only the near infrared band. The idea behind this was that the satellite will shoot a laser beam across the land area and the beam will be reflected back by the most objects, except water. It is one of the only things that can absorb the NIR laser, so the values of the NIR resulted images ,over the water bodies, will be equal to 0. Near the water banks, because the volume of water is too small to absorb all the NIR rays, the pixel values will not be exactly 0 (the value range can vary up to 15-20). The method seemed to work when we considered all the pixel values between 0-15 to be a water part, with a calculated accuracy of around 83% (from our tests done over a set of 12 images), but we encountered a big drawback. On the flatter land areas, lacking valleys and mountains, the images looked pretty good, but when the topography was to diverse the shadows began to appear. The pixel value of the areas covered by shadows can vary between 10 and 25, so we got a lot of false positive cases.

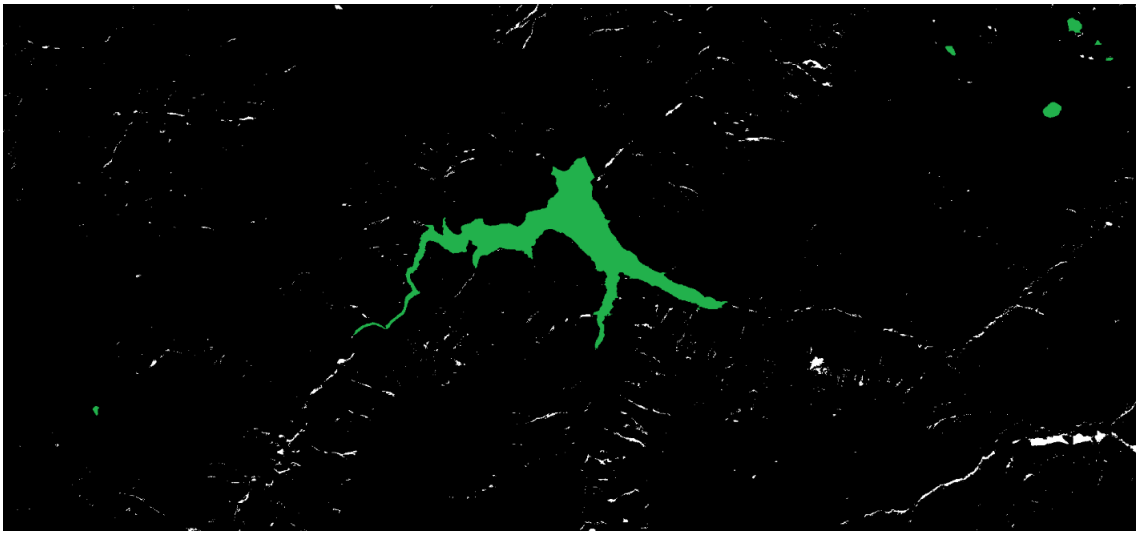


Fig 12 - False positive water detection

We can take a look in the previous picture (Fig 12) to see how significant are the errors caused by the shadows. The green areas represent the water bodies classified correct and the white areas are the shadows that our application classified as water. It may not seem much but that figure is 1414 x 658 pixels in size covering around 930.412 m^2, and it will be a waist of money to move all the people from that areas in case of a flood just because we classified the shadows as water.

The method that we chose in the end was to use a index called NDWI or Normalized difference water index. NDWI is a remote sensing indicator which is obtained by combining the NIR and green band images. The formula will be applied as following for each pixel:

$$NDWI/perpixel = \frac{Xgreen - Xnir}{Xgreen + Xnir}$$

If the calculated value is bigger than 0.45 we can classify that specific pixel as a spot that contains water, and all the other values below that can be classified as non-water pixels (land). This method shown a very satisfying result compared to the true value, with an calculated accuracy of around 88% (the tests were done by us over a set of 12 images; over the oceanic area the accuracy was around 98% and over the more diverse land area the accuracy was around 82%). The big advantage of this approach was that it dose not identify any shadows as water pixels, without losing the accuracy.

There are numerous studies [31, 20] that suggested that NDWI is one of the best classification method, for optical images, comparable with the multifractal formalism method (with an accuracy of 89%), and the neural network classification approach that shown great results in some cases, but it did not provide a proper classification because of its high sensitivity to speckle noise.

In the end, using this approach we manage to detect all the water bodies with a great accuracy with a insignificant false positives like shadows and other objects. We can take a look in the following picture (Fig 13) on how a result image using NDWI would look like. In this picture all the white zones are water areas, and if we make a difference between the Fig 12 and Fig 13 we can see a big improve when we take in consideration the error caused by the shadows.

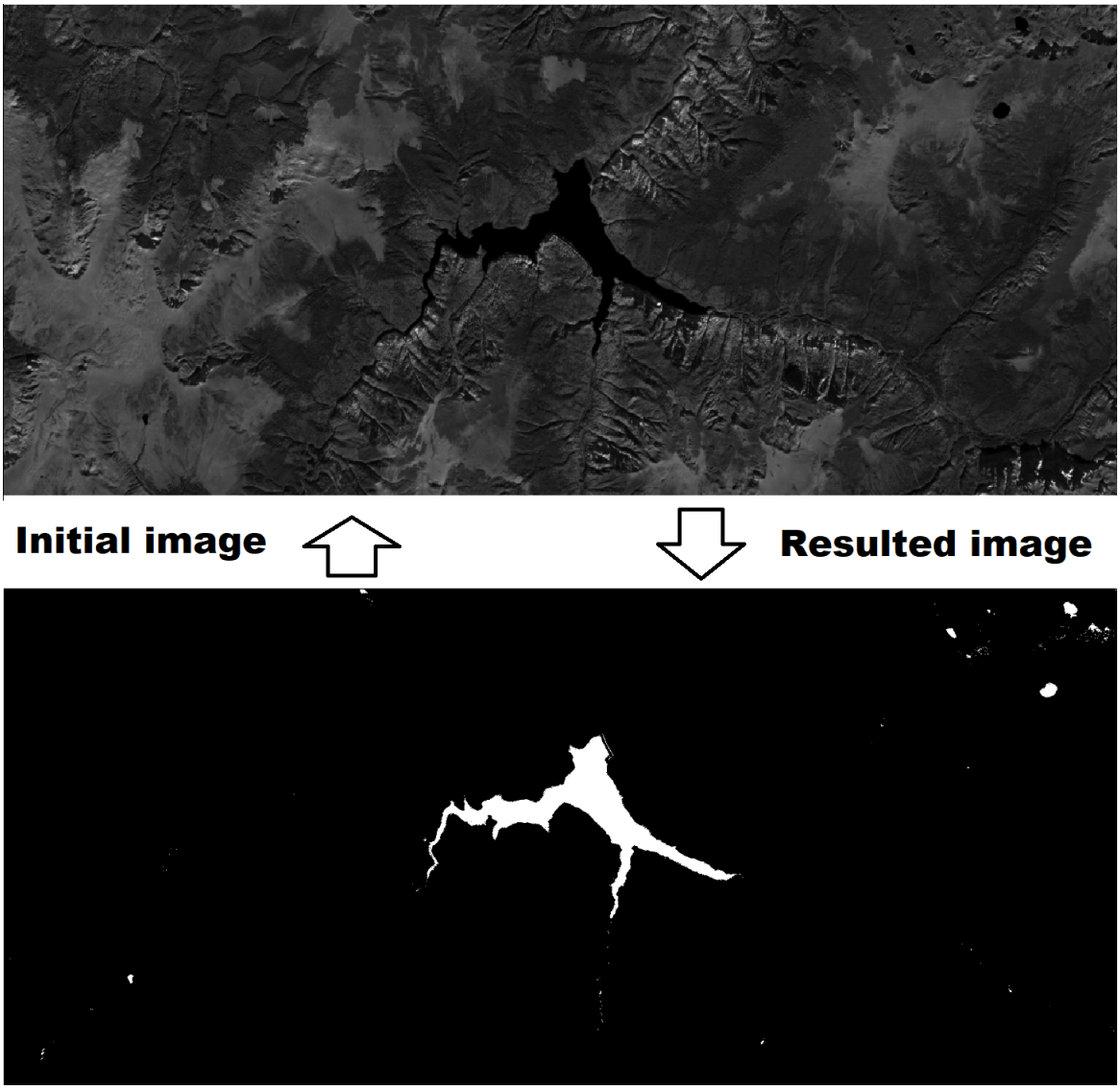


Fig 13 - Water detection using NDWI (white areas are water bodies)

After the application has managed to detect all the water bodies from a image it will need to overlay the processed image over the topographic image. This will be a topic that we will cover in the next subsection.

2.3.2 Mapping the processed images (containing water) to the topographically images

In this section we will discuss about how we managed to map two images (the processed image, containing water, with the topographical image). The idea behind this process is that, usually the topographical images available online tend to cover a very large area,

that can be even 10 times bigger that the image from which we detect the water bodies, so when we apply the flood algorithm that should extend the water areas based on the topographical surface we cannot just simply place the smaller image over the larger one without taking into account the coordinates.

Fortunately for us, every topographical, NIR, and green band image offered both by Landsat and Sentinel has a header that contains the metadata (information about the data; structural, administrative and descriptive) for that specific image. In the header we can find the coordinates of each corner of an image, and we can use the Gdal library to take use of that data. The Gdal library can parse every header offered by satellite images and this is why we chose to use this approach.

We cannot just simply overlay one image above another because the pictures can have different resolutions, some of them have a resolution of 10 m^2 per pixel, while others can have a resolution 30 m^2 per pixel. What we did instead was to create a function that can map every pixel to a global coordinate (of form longitude and latitude), and we can search for that coordinate set in the second image. For every image header we apply a function offered by Gdal called dataset.GetGeoTransform which can parse the metadata, and then based on obtained data a Spatial Reference is created for each picture (Spatial reference is a class offered by Gdal that can reference every satellite image to a place on Earth based on the metadata obtained). The Spatial Reference for both images (our processed image and topographical image) is then passed as an argument together with the transformation header, and the pixel coordinates to a function that will return the longitude and latitude corresponding to the specified pixel.

```
public static double[] getCoordinatesFromPixel(double x, double y, double[] transformation,
                                               SpatialReference spat_ref1, SpatialReference spat_ref2){
    /**
     * @param x,y coordinates from pixel
     * @param transformation dataset. GetGeoTransform contains xoff, a, b, yoff, d, e
     * @return a array of double, first value is longitude and second latitude
     */
    double[] results = new double[2];

    results[0] = transformation[1]*x + transformation[2]*y + transformation[0];
    results[1] = transformation[4]*x + transformation[5]*y + transformation[3];

    CoordinateTransformation t = new CoordinateTransformation(spat_ref1, spat_ref2);
    results = t.TransformPoint(results[0], results[1]);

    return results;
}
```

Fig 14 - Function for obtaining global coordinates from pixel coordinates

There is an another function created by us that can get as an argument the longitude, latitude, transformation dataset and the Spatial References for two images and it can return the pixel coordinates corresponding to the specified global coordinates.

These functions will be used to help us on the flood algorithm that we will discuss about in the next subsection

2.3.3 The flood algorithm applied over the images

In this section we will discuss about how the flood algorithm works. Firstly the water detection class is called, which will combine the NIR and the green band images resulting a processed image that will color all the water bodies with the white color, and the rest (land cover) will be colored black.

After the water is detected, the flood algorithm is called. The algorithm is stored in a class that will take as input parameters the name of the NIR image, the name of the processed image(containing the water bodies), and the name of the topographic image. The only reason that we had to pass as an input parameter the name of the NIR image is because our processed image dose not contain a file header (containing the global coordinates of each corner), but essentially the images are pretty much the same, when it comes to the land cover area, so we had to use the NIR header as the matadata for our processed picture.

The flood algorithm that we created is based on the square flood fill algorithm. The idea is that the image is parsed in a linear manner, and when it encounters a white pixel (containing water) it starts a fill algorithm, that pixel will be colored to green and its based on the topographical map will be retained (the height is obtained by mapping the pixel from our processed image to the topographical map (using the functions described in the previous subsection). Every neighbor (up, down, left, right; this is why is called square flood fill) of that pixel will be place on a stack.

The elements are popped from the stack, one at a time, and every element is checked. If the element is a white pixel (already containing water), it is colored green (we chose to color the already parsed pixels with green to know that they have been checked, so they will not be added again on the stack; it servers as a mark flag), and its neighbors will be added to the stack. If the element is a black pixel (land area) it will be subjected to a test, and if the test is passed the pixel is colored green, and all his neighbor will be pushed on the stack. The test consists on several things.

Firstly the coordinates (set x,y) of the black pixel will be gathered from our processed image. The x,y coordinates will be transformed to a global longitude and latitude. The obtained global coordinates will be mapped to the topographical image such that the algorithm will be able to identify the exact position of our black pixel on the topographical map. The height of our black pixel is than compared to the previous stored height, and if the height difference is small enough, the pixel will pass the test. Besides the height difference we tried to take into account the distance from the river, such that we store the coordinates of the closest river (water) pixel, and we calculate the height difference based on the distance from the water. The reason because we used this extra test case is because of the swampy regions. They tend to be pretty flat, with a small height variation, but we all know that the water cannot extent to the infinity, so this is why we chose to use this extra case. The height difference that we accepted was between $CurrentHeight - 2$ and $CurrentHeight + 2$ (the range was selected by us based on several tests and comparisons with real floods).

In the following pictures (Fig 15, Fig 16) we can take a look on how our application will get through every process (detecting water and applying the flood algorithm). On Fig 15 in the right side we can see how a NIR input image would look like and in the left side is the resulting image after we apply the detection of water bodies.

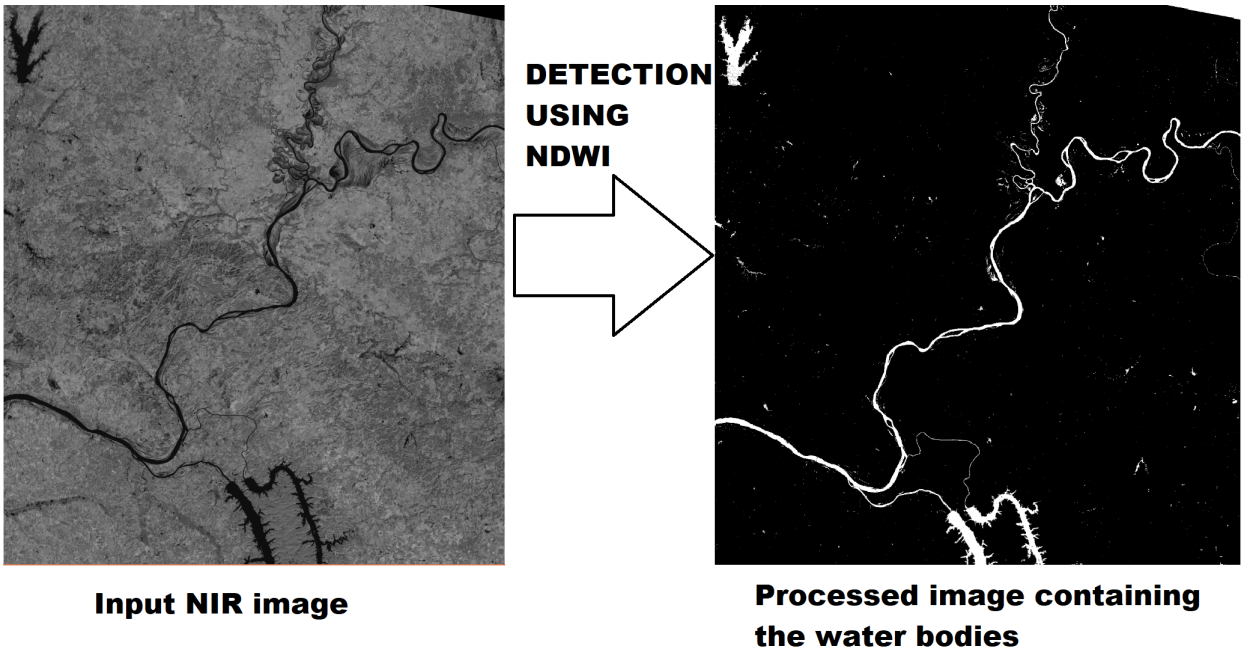


Fig 15 - ...

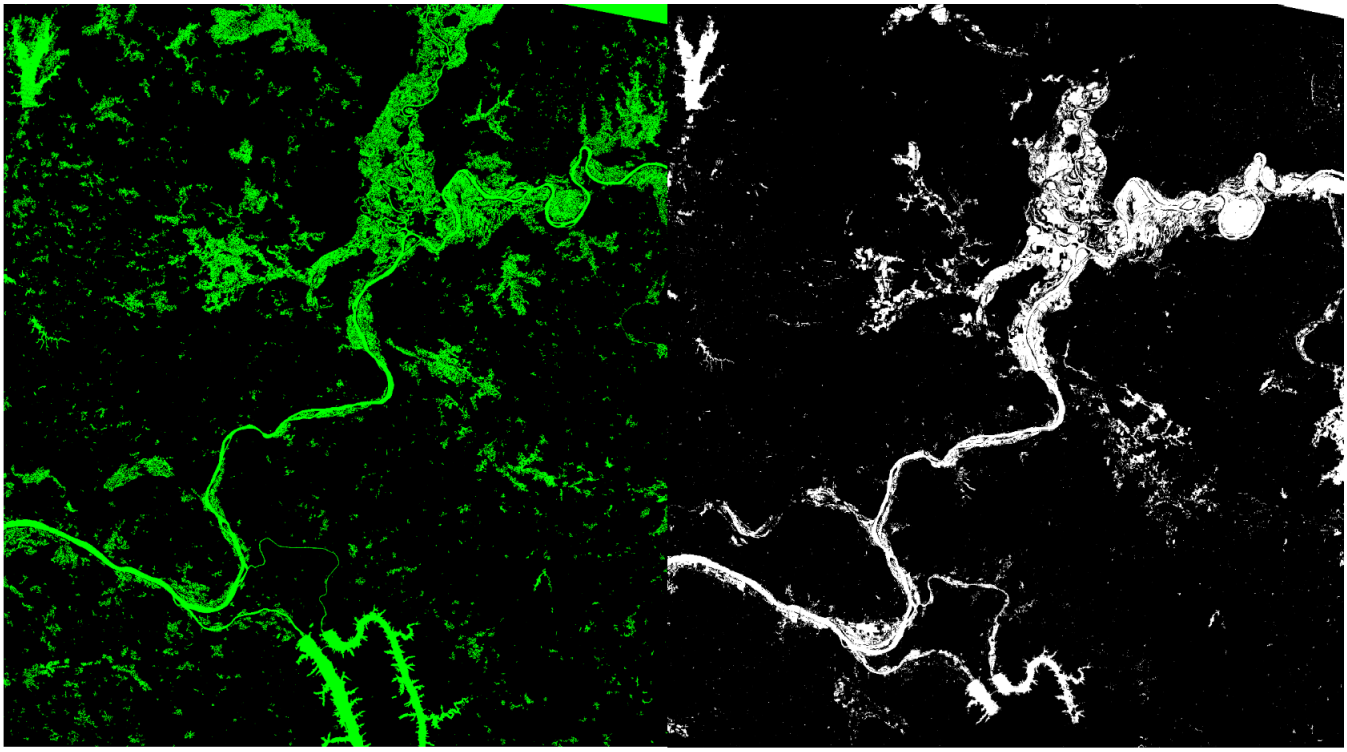


Fig 16 - ... coordinates

In the above picture (Fig 16) we can see, on the left side, how a prediction made by our algorithm would look like. The image is obtained by applying the flood algorithm over the processed image (which contains the water bodies) on the right side of Fig 15.

In the right side of Fig 16 we can see an image of an actual flood area that was captured by the Landsat 7 satellite over a section of Mississippi river in Cairo, Illinois, USA. There are some zones containing water (in the image of the actual flood) that cannot be seen because of the clouds. The green waves cannot penetrate the clouds, meaning that some images cannot be processed correctly because of that problem.

Another possible cause that can make some differences, between the two images, is the fact that our flood algorithm dose not take into account the volume of water dropped over a specific area during a rainstorm. That kind of data can be hard to acquire by a normal user and can make the whole process much expensive (That is what we tried to avoid with our application). After the pictures are being processed, the image sets are archived and an email, containing a link from where the data can be downloaded, is send to the user that had placed the request in the first place.

Chapter 3

The application

3.1 Qgis introduction

3.2 Python GDAL introduction

3.3 Functional description

3.4 The user interface

3.5 Main use cases

3.6 Implementation details

Chapter 4

Conclusions

4.1 Performance Evaluation

4.2 Future Development

Bibliography

[1] Rover J., Ji L., Wylie B.K., Tieszen L.L. Establishing Water Body Areal Extent Trends in Interior Alaska from Multi-Temporal Landsat Data. Remote Sens. Lett. 2012;3:595–604. doi: 10.1080/01431161.2011.643507

[2] Alsdorf D.E., Rodríguez E., Lettenmaier D.P. Measuring Surface Water from Space. Rev. Geophys. 2007;45 doi: 10.1029/2006RG000197. [CrossRef] [Google Scholar]

[3] Flood forecasting - World meteorological organization *http://www.wmo.int/pages/prog/hwrp/FloodForecastingInitiative.php* - accessed in 15.05.2019

[4] Xie, K; Ozbay,K;Zhu, Y;Yang, H. Evacuation zone modeling under climate change: A data-driven method. J.Infrastruct. Syst. 2017,23,04017013

[5] Pitt, M. Learning Lessons from the 2007 Floods; Cabinet Office: London, UK, 2008.

[6] Lohani, A.K.; Goel, N.; Bhatia, K. Improving real time flood forecasting using fuzzy inference system.J. Hydrol. 2014, 509, 25–41

[7] Nayak, P.; Sudheer, K.; Rangan, D.; Ramasastri, K. Short-term flood forecasting with a neurofuzzy model.Water Resour. Res. 2005, 41.

[8] Taherei Ghazvinei, P.; Hassanpour Darvishi, H.; Mosavi, A.; Yusof, K.B.W.; Alizamir, M.; Shamshirband, S.; Chau, K.W. Sugarcane growth prediction based on meteorological parameters using extreme learning machine and artificial neural network. Eng. Appl. Comput. Fluid Mech. 2018, 12, 738–749

[9] Kasiviswanathan, K.; He, J.; Sudheer, K.; Tay, J.-H. Potential application of wavelet neural network ensemble to forecast streamflow for flood management. J. Hydrol. 2016, 536, 161–173.

[10] Ravansalar, M.; Rajaee, T.; Kisi, O. Wavelet-linear genetic programming: A new approach for modeling monthly streamflow. J. Hydrol. 2017, 549, 461–475.

[11] Mosavi, A.; Rabczuk, T. Learning and intelligent optimization for material design innovation. In Learning and Intelligent Optimization; Springer: Cham, Switzerland, 2017; pp. 358–363.

[12] Dandagala, S.; Reddy, M.S.; Murthy, D.S.; Nagaraj, G. Artificial neural networks applications in groundwater hydrology—A review. Artif. Intell. Syst. Mach. Learn. 2017, 9, 182–187.

[13] Deka, P.C. Support vector machine applications in the field of hydrology: A review. Appl. Soft Comput. 2014, 19, 372–386.

[14] Fotovatikhah, F.; Herrera, M.; Shamshirband, S.; Chau, K.-W.; Faizollahzadeh Ardabili, S.; Piran, M.J. Survey of computational intelligence as basis to big flood management: Challenges, research directions and future work. Eng. Appl. Comput. Fluid Mech. 2018, 12, 411–437.

[15] McFeeters, S.K. The use of the normalized difference water index (NDWI) in the delineation of open water features. Int. J. Remote Sens. 1996, 17, 1425–1432.

[16] Duan, Z.; Bastiaanssen, W.G.M. Estimating water volume variations in lakes and reservoirs from four operational satellite altimetry databases and satellite imagery data. Remote Sens. Environ. 2013, 134, 403–416.

[17] Poulin, B.; Davranche, A.; Lefebvre, G. Ecological assessment of phragmites australis wetlands using multi-season spot-5 scenes. Remote Sens. Environ. 2010, 114, 1602–1609..

[18] Hui, F.; Xu, B.; Huang, H.; Yu, Q.; Gong, P. Modelling spatial-temporal change of poyang lake using multitemporal landsat imagery. Int. J. Remote Sens. 2008, 29, 5767–5784.

[19] Multifractal Analysis of Hydrologic Data - Tongzhou Zhao, Liang Wu, Dehua Li, Yiming Ding, Multifractal Analysis of Hydrologic Data Using Wavelet Methods and Fluctuation Analysis, published in Discrete Dynamics in Nature and Society Volume 2017

[20] Comparison between NDWI, Machine Learning, and Multifractal analysis for detecting water bodies - V. M. San Martina, Alejandra Figliolaa, Application of Multifractal Analysis to Segmentation of WaterBodies in Optical and Synthetic Aperture Radar Satellite Images, originally announced April 2016 and published in Cornell University Journal.

[21] Flood forecasting models - S. Nevo, V. Anisimov, G. Elidan, R. El-Yaniv, P. Giencke, Y. Gigi, A. Hassidim, Z. Moshe, M. Schlesinger, G. Shalev, A. Tirumali, A. Wiesel, O. Zlydenko, Y. Matias , ML for Flood Forecasting at Scale originally announced in January 2019, published in 28 January 2019 on arXiv.

[22] ML hydro-dynamic modeling Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Data-driven discretiza-tion: a method for systematic coarse graining of partial differential equations.arXiv preprintarXiv:1808.04930, 2018

[23] USGS *https://earthexplorer.usgs.gov/* - accessed in 19.05.2019

[24] Wikipedia Landsat Program *https://en.wikipedia.org/wiki/Landsat_program* - accessed in 19.05.2019

[25] Copernicus *https://scihub.copernicus.eu/* - accessed in 19.05.2019

[26] QGIS *https://www.qgis.org/en/site/about/index.html* - accessed in 19.05.2019

[27] Landsat-error *https://www.pixalytics.com/landsat-quirks/* - accessed in 22.05.2019

[28] Gdal *https://en.wikipedia.org/wiki/GDAL* -accessed in 22.05.2019

[29] Flask vs Django *http://www.mindfiresolutions.com/blog/2018/05/flask-vs-django/* - accessed in 22.05.2019

[30] Flask server vs Django server *http://blog.gmludo.eu/2015/02/macro-benchmark-with-django-flask-and-asyncio.html* -accessed in 22.05.2019

[31] A. S. Rogers, M. S. Kearney, Reducing signature variability in unmixing coastal marshthematic mapper scenes using spectral indices, International Journal of Remote Sensing25 (12) (2004) 2317–2335.