An automated system for water mapping from satellite data

Shahriar Mahmood
Department of Computer Science and Engineering
American International University Bangladesh (AIUB)
Dhaka, Bangladesh
shahriar1@ymail.com

Md. Abdul Mukib
Lecturer, Department of Computer Science and Engineering
Hamdard University Bangladesh
Munshiganj, Bangladesh
mukib.hub@gmail.com

Abstract—Water mapping is the idea of monitoring the distribution, status and the flow of water resources to identify where the resources are concentrated. Water stress comes when water use in geographic areas outstrips supply and water resources deteriorate or the water concentration level increases in pure land due to flood or any other factors. Experts and Researchers uses the images from satellite for mapping the flow of water concentrations through a country for keeping water flow track and fresh water level track which can also be used in data mining for present and future estimations. Hence, Experts and Researchers may use an automated tool for generating the water map. This will certainly increase the efficiency of their work. This research is to make an automated system which will automatically download MODIS data when new data is available, resample it and then create an NDVI sample for water mapping. The tools are currently working with data of Bangladesh and in future we will look forward to deal with a larger scale.

Keywords-water mapping, GIS, NDVI, GPS, sattelite, GDAL resampling etc.

I. INTRODUCTION

Agriculture is the single largest producing sector of the economy of Bangladesh. Because of Bangladesh's fertile soil and normally ample water supply, rice can be grown and harvested three times a year in many areas. Due to a number of factors, Bangladesh's labor-intensive agriculture has achieved steady increases in food grain production despite the often unfavorable weather conditions. Thus we need to find a way to monitor these natural disasters such as flood, riverbank erosion and flash flood [2]. And we can do this through water mapping. Water Mapping plays a vital role in increasing the efficiency for monitoring the water flow and the depth of the water resources. Thus, analysts can take help from the tool for the future predictions and also for the current condition of water resources across the country. This research basically focusing in the climate change and the industrial effects. As, it is known that 22% of water are used in industrial purposes. Major industrial users include hydroelectric dams, thermoelectric power plants, which use water for cooling, ore and oil refineries, which use water in chemical processes, and manufacturing plants, which use water as a solvent. Water withdrawal can be very high for certain industries, but consumption is though much lower than that of agriculture. Thus, water mapping can be very useful for the better decision making for the industrial analysts. Industries are generally built near natural water resources for enough water access for their purpose. Climate change does have a strong impact on the water resources across the globe because of the close connections between the climate and the hydrological cycle. Rising temperatures will increase evaporation and lead to increases in precipitation, though there will be regional variations in rainfall. Overall, the global supply of freshwater will increase. Both droughts and floods may become more frequent in different regions at different times, and dramatic changes in snowfall and snow melt are expected in mountainous areas [3].

II. Background Study

A. *Water Mapping*

Water mapping is the idea of monitoring the distribution, status and the flow of water resources to identify where the resources are concentrated. Water stress comes when water use in geographic areas outstrips supply and water resources deteriorate or the water concentration level increases in pure land due to flood or any other factors.

The Process uses in water mapping is-

Step 1. Input MODIS storage data

Step 2. Output 3 band files band2, band3 and band7

Step 3. Process NDVI from band2 and band3

Step 4. Output NDVI between range [-1;+1]

Step 5. Input of NDVI is a [-1.0;+1.0] range

Step 6. Input of Band7 is original MODIS storage band7=10000*band7

Step 7. Process data for water mapping

Step 8. Returns 0 if no water is found and 1 if water is found

Some water mapping applications are given below:

1. Monitoring sustainability of different technologies
2. Flood monitoring at national level
3. Planning of new investments and rehabilitation at local government level
4. Monitoring equity of investment in the rural water sector at national level and international level.
5. Monitoring coverage of improved water sources at the national level and international level.

B. *Modis*

MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument aboard the Terra (EOS AM) and Aqua (EOS PM) satellites [1]. Terra's orbit around the Earth is timed so that it passes from north to south across the equator in the morning, while Aqua passes south to north over the equator in the afternoon. Terra MODIS and Aqua MODIS are viewing the entire Earth's surface every 1 to 2 days, acquiring data in 36 spectral bands, or groups of wavelengths (see MODIS Technical Specifications).

C. *NDVI*

The Normalized Difference Vegetation Index (NDVI) is a simple graphical indicator that can be used to analyze remote sensing measurements, typically but not necessarily from a space platform, and assess whether the target being observed contains live green vegetation or not.

$$NDV\,I = (NIR - RED) / (NIR + RED) \qquad (1)$$

Theoretically, NDVI values are represented as a ratio ranging in value from -1to 1 but in practice extreme negative values represent water, values around zero represent bare soil and values over 6 represent dense green vegetation.

D. *GDAL*

GDAL (Geospatial Data Abstraction Library) is a library for reading and writing raster geospatial data formats. Several software programs use the GDAL/OGR libraries to allow them to read and write multiple GIS formats. Such programs include: ArcGIS, Grass GIS, Google Earth [4].

E. *MODIS rerojection tools*

Native MODIS data files are stored in HDF-EOS (Hierarchical Data Format Earth Observing System), a file format that does not currently have wide support. To deal with this problem we need to resample modis data files from HDF format to other formats like .tiff, .jpg, .jpeg etc. to make readable by the available tools. MRT serves this purpose for us.

F. GIS

A geographic information system is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data. The term describes any information system that integrates stores, edits, analyzes, shares, and displays geographic information for informing decision making. A geographic information system (GIS) integrates hardware, software, and data for capturing, managing, analyzing, and displaying all forms of geographically referenced information.

G. TOOLS

This research used several tools to finish the work. They are-
QGIS (Quantum GIS), Python programming (interpreted, object-oriented, high-level programming language with dynamic semantics), GDAL (Geospatial Data Abstraction Library), PHP (recursive acronym for PHP: Hypertext Preprocessor) and MySQL (an open source Relational Database Management System) [6].

III. PROPOSED METHOD FOR WATER MAPPING

As this research is to generate a system that can do water mapping, it requires MODIS data download, Resampling of those data and then generate NDVI images for those data before going to water mapping process.

A. Water Mapping

We are working with MODIS data for water mapping. So we need to download MODIS data of tile h26 v06 in in HDF format from Modis Site: http://modis.gsfc.nasa.gov The diagram of Modis Grid Given(Figure: 4.1). Basically we are taking h26 v06 which actually covers the area of Bangladesh as shown:

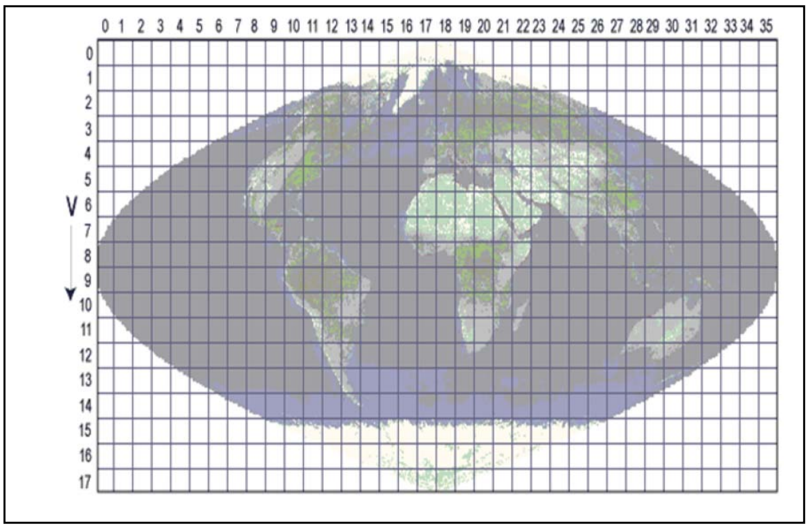
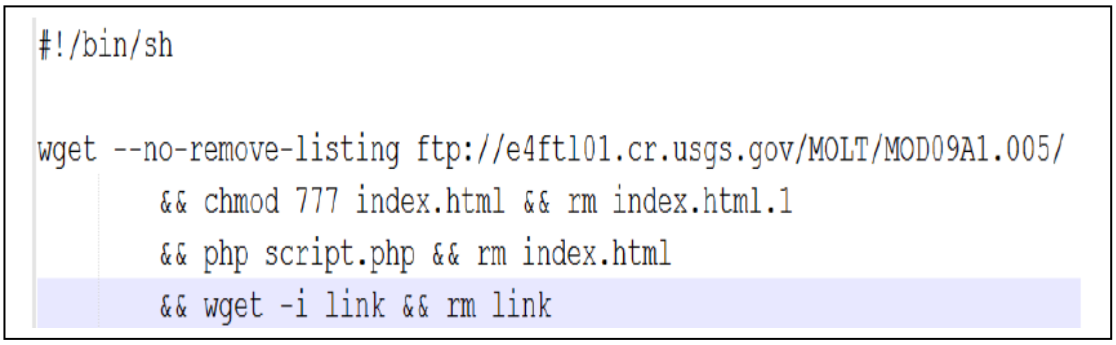


Figure 1: Grid View of World Map

As our objective to build an automated tool so we need to download modis data automatically whenever modis site is updated with new data. Modis site updates with new data once in a week. We have script which always checks modis sites and when sees any new data in modis site the script will automatically download latest modis data and rename the files with the date of the image captured. Then saves the downloaded file into the specific path. The script to automatically download modis data is given below-



```
#!/bin/sh

wget --no-remove-listing ftp://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/
        && chmod 777 index.html && rm index.html.1
        && php script.php && rm index.html
        && wget -i link && rm link
```

Figure 2: Script for downloading MODIS data dynamically

The downloaded modis data files are in HDF format. Most of the current tools does not support HDF format data files. So we need to find a way to convert this HDF files into some other readable formats. This process is named as resampling [5].

B. Resampling

We have used MRT to do resampling. It takes the downloaded HDF files as input and output three bands of the images in Geotiff format. Geotiff format is supported by all current GIS tools. Every image actually has seven bands, but we are generation only three bands, band 2, band 3 and band 7. We need only these three bands for our research. Band2 is Near Infrared (NIR), Band3 is Red and Band7 is Blue.

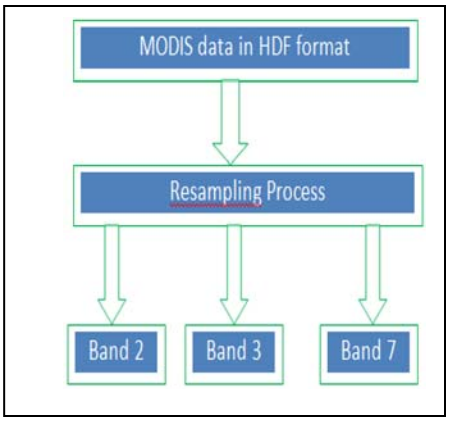


Figure 3: Resampling Process

The resampling code uses in this work is-

```
# Resampling all the hdf files
for j in data/*.hdf
do
     ./resample -p myInput.prm -i $j -o $j.tif
     echo $j  >> name.txt

done
```

Figure 4: Resampling Code

Some sample images after finishing the resampling process are given below-

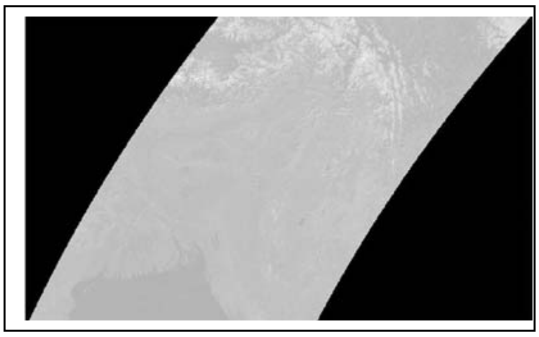


Figure 5: Resampling with Band 2

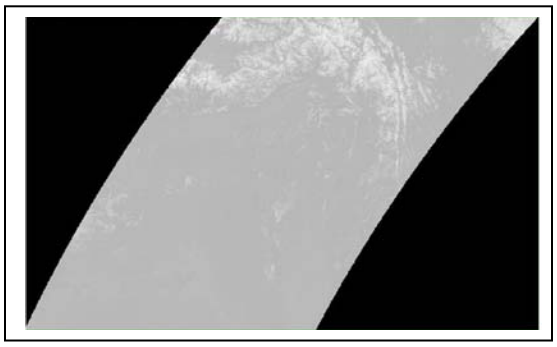


Figure 6: Resampling with Band 3

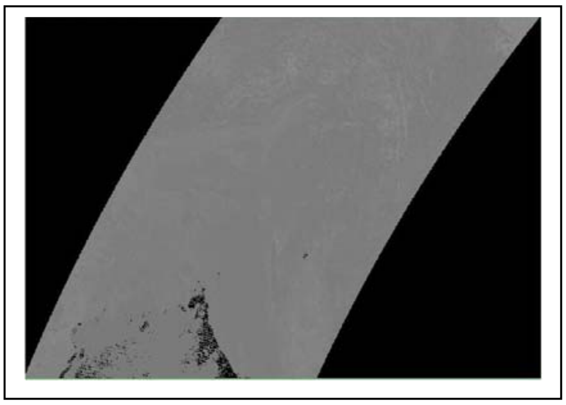


Fig 7: Resampling with Band 7

Now the three bands band2, band3, and band7 is generated in Geotiff format from HDF format. Now it's the time for further processing of these bands for water mapping.

C. NDVI

Our next task is to generate NDVI image. We will use band2, band3 for this process.

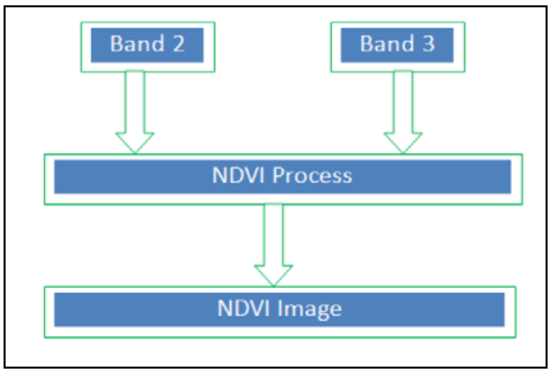


Figure 8: NDVI Processing

Code uses for completing the NDVI operation is given below-

```
#!/usr/bin/python
import wx
import wx.lib.filebrowsebutton as filebrowse
import os
import pp
import time
# For Image Processing
import numpy
from osgeo import gdalnumeric
from osgeo import gdal
from osgeo import gdal_array
from osgeo.gdalconst import *
from multiprocessing import Pool
#from sys import argv
#script, redchanName, nirchanName, ouputName = argv
# Define satellite bands
# Based on Landsat channels
#redchan = redchanName
#nirchan = nirchanName
# Define output file name
#output = ouputName
# Define vegetation indices types
vi_selected = 'NDVI'
# Define Info Message
overview = """Vegetation Index Processing.

Calculates vegetation indices based on biophysical parameters.
NDVI: Normalized Difference Vegetation Index
NDVI = (channel 2 - channel 1) / (channel 2 + channel 1)
NDVI = (NIR - Red) / (NIR + Red)
"""
# Process Equations, Handling and saving of output
def OnOK(redchan, nirchan, output):
   print "red: ", redchan, " nir:",nirchan, " out:", output
   if(redchan==''):
              OnFileInError()
   elif(nirchan==''):
              OnFileInError()
   else:
              redband = gdal_array.LoadFile(redchan)
              nirband = gdal_array.LoadFile(nirchan)
   # NDVI
   result=ndvi(redband, nirband)

   SaveArrayWithGeo(result, redchan,output,'GTiff')
   return 1
def ndvi(redchan, nirchan ):
   """
   Normalized Difference Vegetation Index
   ndvi( redchan, nirchan )
   """
   result = 1.0×( nirchan - redchan )
   result /= 1.0× ( nirchan + redchan + 0.01 )
   result[result<-1.0]=-1.0
   result[result>1.0]=1.0
   return result
def SaveArrayWithGeo(array, src_filename, dst_filename, format ):
   """
   SaveArrayWithGeo(): Saves an Array (array) with Georeferencing from another file (src_flnm),
          save it in file (dst_flnm) with format (format)
   SaveArrayWithGeo (self, array, src_filename, dst_filename, format)
   """
   # Read information from source file.
   src_ds = gdal.Open(str(src_filename))
   gt = src_ds.GetGeoTransform()
   pj = src_ds.GetProjection()
   src_ds = None
   # Create GDAL dataset for array, and set georeferencing.
   src_ds = gdal_array.OpenArray( array )
```

```
    src_ds.SetGeoTransform( gt )
    src_ds.SetProjection( pj )
    # Write array dataset to new file.
    driver = gdal.GetDriverByName( format )
    if driver is None:
            raise ValueError, "SaveArrayWithGeo: Can't find driver "+format
    return driver.CreateCopy( dst_filename, src_ds )
"""
if __name__ == '__main__':
    pool = Pool(processes=4)
    result = pool.apply_async(OnOK, [])
    print result.get ()
"""
for line in open('name.txt','r'):
    mylist = []
    for item in line.split():
        mylist.append(item)
        band2File = mylist[0] + '.sur_refl_b02.tif'
        band3File = mylist[0] + '.sur_refl_b03.tif'
        outputFile = mylist[0] + '.ndvi.tif'
        #print item
        #import sys
        #sys.exit()
        OnOK(band2File, band3File, outputFile)
#OnOK()
#ppservers = ()
#job_server = pp.Server(ppservers=ppservers)
#job_server.submit(OnOK, (), (ndvi,),('numpy','os'))
 #OnOK(mylist[0], mylist[1], mylist[2])
#job_server.wait()
```

Calculates vegetation indexes based on biophysical parameters. NDVI: Normalized Difference Vegetation Index

NDVI = (channel2 - channel1) / (channel2 + channel1) (2)

NDVI = (NIR - Red) / (NIR + Red) (3)

From the code it can be sate that,

NDV I = 1.0 × (nirchan - redchan) / (nirchar + redchan + 0.01) (4)

Here, Nirchan = Near Infrared (Band2)

Redchan = Red (Band2)

Multiplying the result with 1.0 to remain the result in floating point and adding 0.01 with the denominator to ensure that the denominator never become zero.

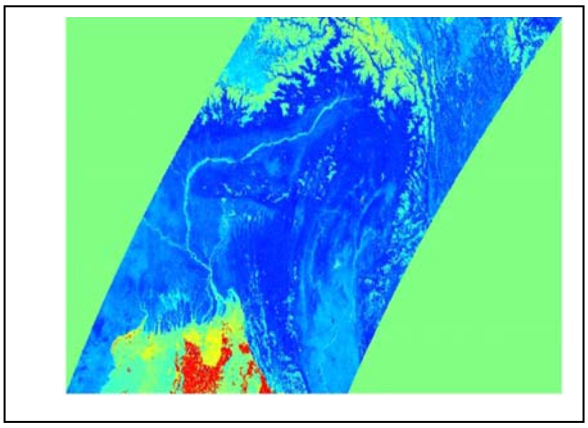


Figure 9: Resultant NDVI

From the image we can see that RED= FOREST or HIGH DENSITY OF LIVE GREEN, GREEN=LIVE GREEN, BLUE= LAND/SURFACE, SKY BLUE= WATER this NDVI Image will be used to process desired water mapping image. Now the NDVI image and band 7 is achieved. Next it is required to process this two to generate desired water mapping image using water mapping algorithm. There are different types of water mapping algorithms are available. They actually almost same in operation, the difference is that which band they are using. This work is using band 2, band 3 and band 7.

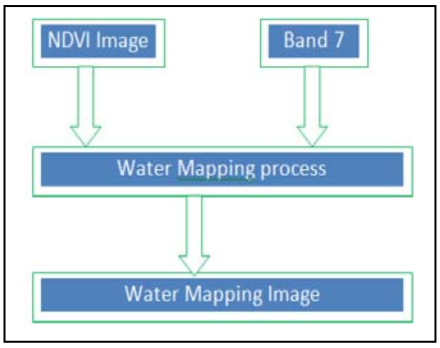


Figure 10: Water Mapping Process

The water mapping algorithm is written below:

```
Step1: Input MODIS storage data.
Step2: Output 3 band files (band2, band3 and band7).
Step3: Process NDVI from band2 and band3.
Step4: Output NDVI between range [-1; +1].
Step5: Input of NDVI is a [-1.0; +1.0] range.
Step6: Input of Band7 is original MODIS storage band 7=10000×band7.
Step8: Process these data for water mapping.
Step9: Returns 0 if no water is found and 1 if water is found.
```

IV. Implementation and Results

As per the water mapping method, after resampling and generating NDVI image file, now the implementation of water mapping is ready to process. The code to implement the water mapping is given below-

```
#!/usr/bin/python
import wx
import wx.lib.filebrowsebutton as filebrowse
import os
import pp
import time
import sys
# For Image Processing
import numpy
from osgeo import gdalnumeric
from osgeo import gdal
from osgeo import gdal_array
from osgeo.gdalconst import *
from multiprocessing import Pool
from sys import argv
#script, ndviFile, band7File, outputFile = argv
# Define satellite bands
# Based on Landsat channels
#ndvi = ndviFile # 'ndvi.tiff'
#band7 = band7File # 'MOD09A1.A2012065.h26v06.005.2012075042721.hdf.sur_refl_b07.tif'
# Define output file name
#output = outputFile # 'water.tif'
# Define Info Message
overview = """Water Mapping
Calculates Water ponding areas based on NDVI and Modis Band 7
Input of NDVI is a [-1.0; +1.0] range
Input of Band7 is original Modis storage band7=10000*band7
Returns 0 if no water is found and 1 if water is found
Xiao X., Boles S., Liu J., Zhuang D., Frokling S., Li C., Salas W., Moore III B. (2005).
Mapping paddy rice agriculture in southern China using multi-temporal MODIS images.
Remote Sensing of Environment 95:480-492.
```

```
Roy D.P., Jin Y., Lewis P.E., Justice C.O. (2005).
Prototyping a global algorithm for systematic fire-affected area mapping using MODIS time series data.
 Remote Sensing of Environment 97:137-162.
"""
# Process Equations, Handling and saving of output
        def OnOK(ndvi, band7, output, GA_Update):
        #print "ndvi: ", ndvi
        #print "band7:", band7
        #print "out:", output
        if(ndvi==''):
                OnFileInError()
        elif(band7==''):
                OnFileInError()
        else:
                ndviF = gdal.Open(ndvi)
                band7F = gdal.Open(band7)
                bndvi = ndviF.GetRasterBand(1)
                bb7 = band7F.GetRasterBand(1)
                test = gdal.Open(ndvi)
                CrAr( ndvi, output, 'GTiff' )
                result = gdal.Open(output, GA_Update)
                for y in range(bndvi.YSize - 1, -1, -1):
                   #Fast way if same pixel sizes...
                   #nxarray=array[j,:].astype(numpy.float32)
                   #nxarray=numpy.where(nxarray>0,water(scanline1, scanline2)
                   scanline1=1.0×bndvi.ReadAsArray(0, y, bndvi.XSize, 1, bndvi.XSize, 1)
                   scanline2=1.0×bb7.ReadAsArray(0, y, bb7.XSize, 1, bb7.XSize, 1)
                   for x in range(0, bndvi.XSize - 1, 1):
                           pix1 = scanline1[0][x]
                           pix2 = scanline2[0][x]
                           scanline1[0][x]=water(pix1, pix2)
        result.GetRasterBand(1).WriteArray(numpy.reshape(scanline1,(1,bndvi.XSize)), 0, y)
        return 1
        def CrAr( src_flnm, dst_flnm, format ):
        """
        CrAr(): Create Array with Georeferencing from another file (src_flnm), save it in file (dst_flnm) with
        format (format)
        CrAr(  src_flnm, dst_flnm, format )
        """
        cr_opts=[]
        # Read information from source file.
        src_ds = gdal.Open(str(src_flnm))
        gt = src_ds.GetGeoTransform()
        pj = src_ds.GetProjection()
        src_ds = None
        # Standard checking on the GDAL driver
        Driver = gdal.GetDriverByName( str(format) )
        if Driver is None:
                raise ValueError, "CrAr: No DriverFound "+format
        DriverMTD = Driver.GetMetadata()
        if not DriverMTD.has_key('DCAP_CREATE'):
                print 'Format Driver %s does not support creation and piecewise writing.
                \nPlease select a format that does, such as GTiff or HFA (Erdas/Imagine).' % format
                sys.exit( 1 )
        # Set up the band number
        nbands = 1
        #print "nbands =", nbands
        # Collect information on source files
        flinfos = names_to_fileinfos( str(src_flnm) )
        ulx = flinfos[0].ulx
        uly = flinfos[0].uly
        lrx = flinfos[0].lrx
        lry = flinfos[0].lry
        # get largest extends
        for fi in flinfos:
                ulx = min(ulx, fi.ulx)
                uly = max(uly, fi.uly)
                lrx = max(lrx, fi.lrx)
                lry = min(lry, fi.lry)
        # Set other info
        psize_x = flinfos[0].geotransform[1]
        psize_y = flinfos[0].geotransform[5]
        band_type = flinfos[0].band_type
        # Try opening as an existing file
        gdal.PushErrorHandler( 'CPLQuietErrorHandler' )
        out_fh = gdal.Open( str(dst_flnm), gdal.GA_Update )
        gdal.PopErrorHandler()
        # Otherwise create a new file
        if out_fh is None:
                geot = [ulx, psize_x, 0, uly, 0, psize_y]
                print geot[0], geot[1], geot[2], geot[3], geot[4]
                xsize = int((lrx-ulx)/geot[1]+0.5)
                ysize = int((lry-uly)/geot[5]+0.5)
        out_fh=Driver.Create(str(dst_flnm), xsize,ysize,nbands,band_type,cr_opts)
                if out_fh is None:
                        raise ValueError, "CrAr: Failed to create new file "+dst_flnm
        sys.exit( 1 )
        out_fh.SetGeoTransform( gt )
        out_fh.SetProjection( pj )
        #out_fh.GetRasterBand(1).SetRasterColorTable(flinfos[0].ct)
        nodata = None
        iband = 1
        for fi in flinfos:
                fi.copy_into( out_fh, 1, iband, nodata )
                iband=iband+1
        iband = 0
def names_to_fileinfos(name ):
        file_infos = []
        fi = file_info()
        if fi.init_from_name( name ) == 1:
                file_infos.append( fi )
        return file_infos
def water (ndvi, band7):
        """
        Water Mapping
        water (ndvi, band7)
        """
        if ((ndvi<0.1) and(band7<400)):
                result=1
        else :
                result=0
        return result
class file_info:
        """A class holding information about a GDAL file."""
        def init_from_name(self, filename):
                """
                Initialize file_info from filename
                filename -- Name of file to read.
                Returns 1 on success or 0 if the file can't be opened.
                """
```

```
fh = gdal.Open( str(filename) )
if fh is None:
return 0
self.filename = filename
self.bands = fh.RasterCount
self.xsize = fh.RasterXSize
self.ysize = fh.RasterYSize
self.band_type = fh.GetRasterBand(1).DataType
self.projection = fh.GetProjection()
self.geotransform = fh.GetGeoTransform()
self.ulx = self.geotransform[0]
self.uly = self.geotransform[3]
self.lrx = self.ulx + self.geotransform[1] * self.xsize
self.lry = self.uly + self.geotransform[5] * self.ysize
ct = fh.GetRasterBand(1).GetRasterColorTable()
if ct is not None:
        self.ct = ct.Clone()
else:
        self.ct = None
return 1
def copy_into( self, t_fh, s_band = 1, t_band = 1, nodata_arg=None ):
"""
Copy this files image into target file.
"""
t_geotransform = t_fh.GetGeoTransform()
t_ulx = t_geotransform[0]
t_uly = t_geotransform[3]
t_lrx = t_geotransform[0] + t_fh.RasterXSize * t_geotransform[1]
t_lry = t_geotransform[3] + t_fh.RasterYSize * t_geotransform[5]

# figure out intersection region
tgw_ulx = max(t_ulx,self.ulx)
tgw_lrx = min(t_lrx,self.lrx)
if t_geotransform[5] < 0:
        tgw_uly = min(t_uly,self.uly)
        tgw_lry = max(t_lry,self.lry)
else:
        tgw_uly = max(t_uly,self.uly)
        tgw_lry = min(t_lry,self.lry)
# do they even intersect?
if tgw_ulx >= tgw_lrx:
return 1
if t_geotransform[5] < 0 and tgw_uly <= tgw_lry:
return 1
if t_geotransform[5] > 0 and tgw_uly >= tgw_lry:
return 1
# compute target window in pixel coordinates.
tw_xoff = int((tgw_ulx - t_geotransform[0]) / t_geotransform[1] + 0.1)
tw_yoff = int((tgw_uly - t_geotransform[3]) / t_geotransform[5] + 0.1)
tw_xsize = int((tgw_lrx-t_geotransform[0])/t_geotransform[1] + 0.5) - tw_xoff
tw_ysize = int((tgw_lry-t_geotransform[3])/t_geotransform[5] + 0.5) - tw_yoff
if tw_xsize < 1 or tw_ysize < 1:
return 1
# Compute source window in pixel coordinates.
sw_xoff = int((tgw_ulx - self.geotransform[0]) / self.geotransform[1])
sw_yoff = int((tgw_uly - self.geotransform[3]) / self.geotransform[5])
sw_xsize = int((tgw_lrx - self.geotransform[0]) / self.geotransform[1] + 0.5) - sw_xoff
sw_ysize = int((tgw_lry - self.geotransform[3]) / self.geotransform[5] + 0.5) - sw_yoff
if sw_xsize < 1 or sw_ysize < 1:
return 1
```

```
                # Open the source file, and copy the selected region.
                s_fh = gdal.Open( str(self.filename) )
                return self.raster_copy( s_fh, sw_xoff, sw_yoff, sw_xsize, sw_ysize, s_band, t_fh, tw_xoff,
                tw_yoff, tw_xsize, tw_ysize, t_band, nodata_arg )
                def raster_copy( self, s_fh, s_xoff, s_yoff, s_xsize, s_ysize, s_band_n, t_fh, t_xoff, t_yoff,
                t_xsize, t_ysize, t_band_n, nodata=None ):
                if nodata is not None:
                        return self.raster_copy_with_nodata(
                        s_fh, s_xoff, s_yoff, s_xsize, s_ysize, s_band_n,
                        t_fh, t_xoff, t_yoff, t_xsize, t_ysize, t_band_n,
                        nodata )
                s_band = s_fh.GetRasterBand( s_band_n )
                t_band = t_fh.GetRasterBand( t_band_n )
                data = s_band.ReadRaster( s_xoff, s_yoff, s_xsize, s_ysize, t_xsize, t_ysize, t_band.DataType)
                t_band.WriteRaster( t_xoff, t_yoff, t_xsize, t_ysize, data, t_xsize, t_ysize, t_band.DataType )
                return 0
        def raster_copy_with_nodata( self, s_fh, s_xoff, s_yoff, s_xsize, s_ysize, s_band_n,t_fh, t_xoff, t_yoff,
        t_xsize, t_ysize, t_band_n, nodata ):
                import Numeric as Num
                s_band = s_fh.GetRasterBand( s_band_n )
                t_band = t_fh.GetRasterBand( t_band_n )
                data_src = s_band.ReadAsArray( s_xoff, s_yoff, s_xsize, s_ysize, t_xsize, t_ysize )
                data_dst = t_band.ReadAsArray( t_xoff, t_yoff, t_xsize, t_ysize )
                nodata_test = Num.equal(data_src,nodata)
                to_write = Num.choose(nodata_test, (data_src, data_dst))
                t_band.WriteArray( to_write, t_xoff, t_yoff )
                return 0
""" Fisrt parallel way -
print "\n::: Starting Water Mapping :::\n"
ppservers = ()
job_server = pp.Server(ppservers=ppservers)
start_time = time.time()
result = job_server.submit(OnOK, (ndvi,band7,output,GA_Update),(CrAr,water, names_to_fileinfos, file_info,
),('gdal','numpy'))
r1 = result ()
job_server.wait()
# Print the partial sum
#print r1
print "\n::: Successfully Finished Water Mapping:::\n"
job_server.print_stats ()
"""
"""2nd Parallel way -
if __name__ == '__main__':
    pool = Pool(processes=4)
    result = pool.apply_async(OnOK, [])
    print result.get()
"""
#OnOK()
#ppservers = ()
#job_server = pp.Server(ppservers=ppservers)
""""""
from multiprocessing import Process
for line in open('resample/testaaaaa.txt','r'):
    mylist = []
    for item in line.split():
        mylist.append(item)
    print mylist
    if __name__ == '__main__':
        p = Process(target=OnOK, args=(mylist[0],mylist[1],mylist[2]))
        p.start()
        p.join()
    #job_server.submit(OnOK, (mylist[0],mylist[1],mylist[2]), (water,CrAr),('numpy','os'))
    #OnOK(mylist[0], mylist[1], mylist[2])
#job_server.wait()
"""
print "\n::: Starting Water Mapping :::\n"
#ppservers = ()
#job_server = pp.Server(ppservers=ppservers)
start_time = time.time()
for line in open('name.txt','r'):
    mylist = []
    for item in line.split():
        mylist.append(item)
        ndviFile = mylist[0] + '.ndvi.tif'
        band7File = mylist[0] + '.sur_refl_b07.tif'
        outputFile = mylist[0] + '.water.tif'
        OnOK(ndviFile, band7File, outputFile, GA_Update)
#result = job_server.submit(OnOK, (ndviFile,band7File,outputFile,GA_Update),
(CrAr,water, names_to_fileinfos,file_info), ('gdal','numpy'))
#job_server.wait()
# Print the partial sum
#print result
print "\n::: Successfully Finished Water Mapping:::\n"
#job_server.print_stats()
```

This codes will process NDVI image and Band 7 and output Water mapping image.

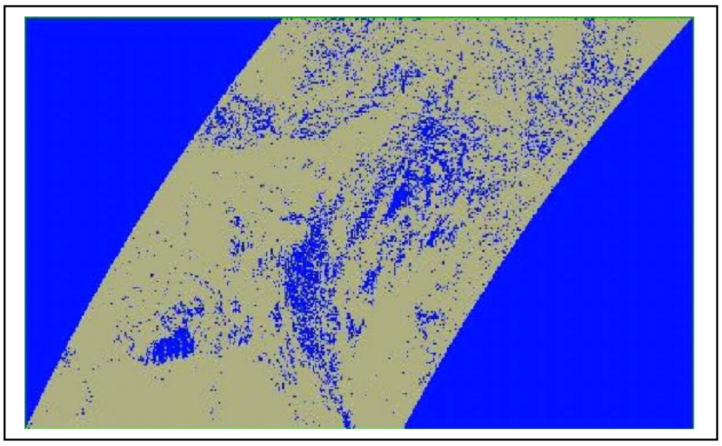


Figure 11: Water Mapping Image - Blue =Water and Gray = Land/ Surface

Now there are some water mapping images of different date are given to compare among them. From the above water mapping images, we can see the differences of water flow in different dates. This images can be used by research and analysts for their future works.

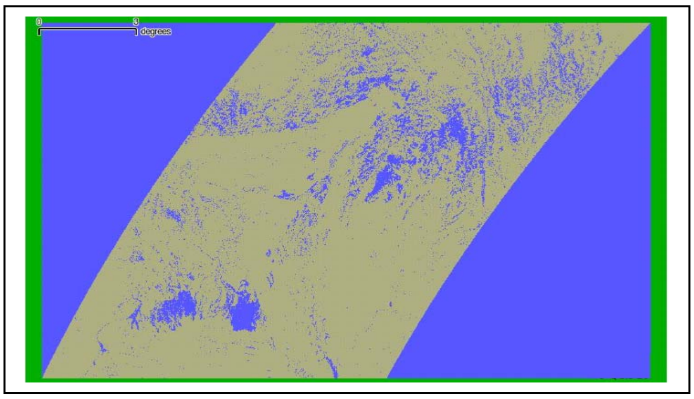


Figure 12: Water Map of date: 2017.02.10

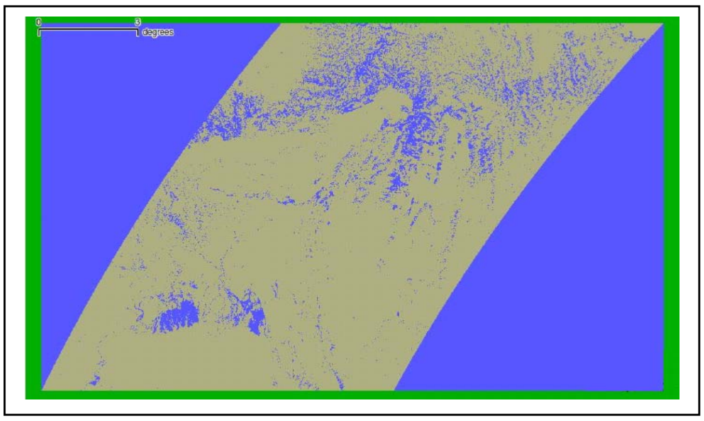


Figure 13: Water Map of date 2017.03.15

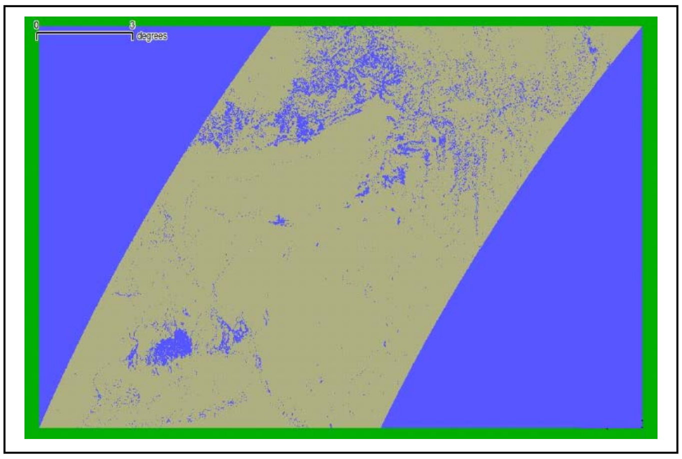


Figure 14: Water Map of date 2017.04.12

From the above figures it is clear that, this work is successful to do the water mapping by using the water mapping algorithm. This images can show the flow of water and can help the researchers to do their research in this field.

V. Conclusion and Future work

This is a very helpful research to find the water mapping in Bangladesh. It allows analysts and users for data mining purposes and allows user a faster way to view this water mapping. Thus, saving a lots of time for users, this simply do not need to download the modis data and use these for water mapping. The document gives the detailed description of the both functional and non-functional requirements for this software development. It also allows analysts and users to get the actual data that can also be directly downloaded from the actual MODIS site. This tool can be used for water mapping the whole ASIA or even a desired region. Coloring of water mapping can be done. Parallel programming can be done with other languages like openmi, mpi for comparing. In this research more parallelism with openmpi, mpi and other programs could be done and compared to use the best one. Also a web portal can be made with the information of this research work so that researchers can get their desired data from anywhere in the world. Researchers can get the information of water mapping of the implemented area from anywhere by the help of the web portal.

Acknowledgment

This research took help from several previous research works. A special thanks to Juan P Guerschman for the work titled “MODIS-based standing water detection for flood and large reservoir mapping [electronic resource]: algorithm development and applications for the Australian continent” and also thanks to B. Schraven for the work titled “Constructing Epistemic Landscapes: Methods of GIS-Based Mapping. The IUP Journal of Knowledge Management”. Also several research work helped to finish this research.

References

[1] J. P. Guerschman, G. Warren, G. Byrne, L. Lymburner, N. Mueller and A. V. Dijk, "MODIS-based standing water detection for flood and large reservoir mapping [electronic resource] : algorithm development and applications for the Australian continent" MARCH 11, 2011

[2] Apel, H. Hung, N.G. Thoss and H. Schne, "GPS buoys for stage monitoring of large rivers." Journal of Hydrology, vol.412-413(0): pp. 182-192, 2012

[3] Delgado, J.M. Merz, and B. Apel, "A climate-flood link for the lower Mekong River." Hydrol. Earth Syst. Sci., Vol.16(5), pp. 1533-1541, 2012

[4] D. Minkwitz, J. Beckheinrich, and A. Hornbostel "Installation of an experimental Galileo GBAS at research Port Rostock", European Journal of Navigation, Vol. 10, No. 1, pp. 16-28, April 2012.

[5] T. Menkhoff, S. Gerke, and H. D. Evers "Water and Knowledge Management in Vietnam: Understanding the Mekong Basin", Social Space (Lien Center for Social Innovation), pp. 74-79.

[6] S. Gebhardt, T. Wehrmann, V. Klinger,I. Schettler,J. Huth, Kuenzer, Claudia und D. Stefan "Improving data management and dissemination in web based information systems by semantic enrichment of descriptive data aspects.", Computers & Geosciences, vol.36 (10), Seiten 1362-1373, Elsevier.