

RELATÓRIO TÉCNICO-CIENTÍFICO FINAL

1. IDENTIFICAÇÃO

Estudante: Neemias Vitorino Freitas

Campus: Campos-Centro

Modalidade da bolsa: PIBITI/IFF

Núcleo de Pesquisa: Núcleo Sistemas de Informação

Área/Linha de Pesquisa: Aeroespacial/Geoprocessamento

Nome do orientador: Fernando Luiz de Carvalho e Silva

Campus: Campos-Centro

2. TÍTULO

Estudar álgebra espectral em fenômenos regionais observáveis por sensoriamento remoto

3. RESUMO

Ao se utilizar de técnicas e métodos de Geoprocessamento e Sensoriamento Remoto, o projeto visa à criação de ferramentas computacionais que permitam a investigação de fenômenos de interesse regional, especialmente problemas locais que se relacionam ao uso do solo, plantio de cultivares como cana-de-açúcar e abacaxi, observados volumes dos corpos lânticos e lóticos para identificar o impacto de ações antrópicas no seu manejo. Foram desenvolvidos dois algoritmos na linguagem de programação Python que manipulam cenas de satélite da região de interesse e retornam uma nova cena com os parâmetros definidos pelo usuário, permitindo, assim, a investigação de fenômenos. No entanto, houve tempo somente para implementação deste em 'linha de código', ou seja, sem interface. Futuramente, é possível a implementação dos algoritmos como plugin de algum software de Sistema de Informação Geográfica, como o Quantum GIS, também estudado neste projeto.

Palavras-Chave: geoprocessamento, sensoriamento, álgebra

4. INTRODUÇÃO

Durante muito tempo, a manipulação de dados geográficos era feita através de mapas e outros documentos impressos ou desenhados em uma base, o que impunha algumas limitações à precisão, ao transporte da informação, à perpetuação ao longo do tempo e à transferência da informação. Com a maior disponibilidade de recursos de Tecnologia de Informação, os dados geográficos passaram a ser tratados por um conjunto de técnicas e ferramentas matemáticas e computacionais, dando suporte às inúmeras áreas e ramos da Geografia, como Geoprocessamento, Sistemas de Informação Geográfica (SIG) e Sensoriamento Remoto (IBGE, 2013). De forma específica, este último, captura informações sobre um alvo na superfície terrestre através de dispositivos orbitais que detectam a energia eletromagnética refletida por esses. Eles apresentam três características básicas principais: a resolução temporal, a espacial e a espectral. A resolução temporal se refere ao intervalo de tempo amostrado do alvo em terra, por exemplo. A resolução espacial descreve a sensibilidade do satélite para identificar alvos em suas dimensões espaciais. A resolução espectral é definida pelo conjunto de sensores diferentes oferecidos, onde cada sensor captura um espectro específico de radiação eletromagnética, tornando o conjunto de cenas, chamado de bandas espectrais, útil para

um grande número de investigações. Desta forma, a investigação de fenômenos atuais e passados, envolve a escolha de bandas espectrais e a definição do processamento a ser realizado através de modelos matemáticos para cálculo dos dados sensoreados, formando assim o conceito de álgebra espectral. Um exemplo do uso desta tecnologia é o *Normalized Difference Vegetation Index* ou Índice de Vegetação por Diferença Normalizada (NDVI) (ROUSE, 1973). Ao se utilizar dessas técnicas e métodos, o projeto visa à criação de ferramentas computacionais que permitam a investigação de fenômenos de interesse regional, especialmente problemas locais que se relacionam ao uso do solo, plantio de cultivares como cana-de-açúcar e abacaxi, observados volumes dos corpos lênticos e lóticos para identificar o impacto de ações antrópicas no seu manejo.

5. OBJETIVOS

a. Geral

Investigar fenômenos de interesse regional, observáveis através da álgebra de bandas espectrais obtidas através de sensoriamento remoto.

b. Específicos

- Capacitar alunos e pesquisadores para o uso das imagens de satélite para resolver problemas locais;
- Identificar uso para as imagens obtidas através das estações terrestres a serem construídas no IFF a partir do Centro de Referência em Sistemas Embarcados e Aeroespaciais;
- Documentar as técnicas utilizadas e as adaptações para as condições locais;
- Formar mão de obra de tecnologia da informação habilitada para trabalhar em conjunto com pesquisadores nas áreas ambiental, energética, defesa civil e meteorológica.

6. METODOLOGIA

Primeiramente foi estudado o conteúdo técnico relacionado à Geoprocessamento, Sensoriamento Remoto e Cartografia provenientes de monografias, teses de mestrado e doutorado, além de estudos dirigidos de universidades da região e do exterior, disponíveis na internet. A todo tempo da pesquisa, foi elaborado um documento com todas as referências bibliográficas encontradas, a fim de que não se perdesse a linha de pesquisa, nem dados úteis que pudessem ser revisados/recuperados posteriormente. Além deste documento, também foram elaborados outros três tutoriais em formato de apresentação de slides Google Docs, que resumem e filtram todo o conteúdo teórico estudado especificamente para este projeto, a fim de que outros bolsistas posteriores ganhem tempo para dar continuidade à pesquisa e familiarizem-se mais facilmente com a área. Prosseguindo com a pesquisa, foi realizado um estudo dirigido de geoprocessamento com a linguagem de programação Python utilizando-se de softwares livres de Sistemas de Informação Geográfica (Quantum GIS, particularmente para este projeto), com a direção do professor Chris Garrard, da Universidade do Estado de Utah (Utah State University, em inglês). A partir desse estudo, foi possível desenvolver dois algoritmos em Python, um para leitura de cenas de satélite e outro para a escrita de novas cenas a partir da leitura e manipulação de outras, permitindo a interpretação de fenômenos na região da cena.

7. RESULTADOS E DISCUSSÕES

Como resultado direto do estudo do conteúdo teórico, foram elaborados três tutoriais em formato de apresentação de slides Google Docs, com os temas: a) Introdução a geoprocessamento; b) Quantum GIS - Introdução; e c) Quantum GIS - Camadas Vetoriais. Eles estão disponíveis publicamente na web e podem ser acessados através dos endereços URL expressos no tópico “Referências Bibliográficas” deste relatório. As imagens a seguir mostram alguns slides dos tutoriais:



Conceitos iniciais

Este tutorial introduz brevemente o software Quantum GIS, abordando superficialmente conceitos de geoprocessamento. No entanto, é altamente recomendável um estudo mais conciso nesta área da geografia para melhor aproveitamento e entendimento do software. Sugerimos este tutorial de *Introdução a geoprocessamento*: bit.ly/105qFwb

ESRI SHAPEFILE

Um shapefile é um formato de arquivo que consiste em um arquivo principal (.shp), um arquivo de índice (.shx), e uma tabela dBASE (.dbf), onde:

- .shp - arquivo contendo feições geométricas.
- .dbf - arquivo contendo os atributos em formato dBase.
- .shx - arquivo índice.

O shapefile é o formato padrão de salvamento das camadas vetoriais feitas tanto no Quantum GIS quanto em outros softwares de SIG. Sem esses três arquivos, não há uma camada vetorial completa.

Quantum GIS


Camadas vetoriais

Criando camadas vetoriais

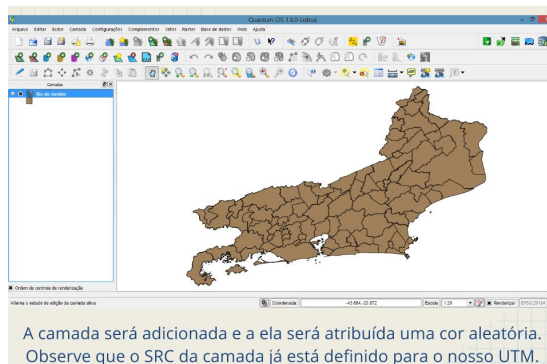
Como criar uma camada vetorial de ponto, linha e polígono

Criando camadas vetoriais

Neste tutorial estarão descritos os procedimentos para criar uma camada vetorial de pontos, linhas ou polígonos. Antes de começar, é recomendável que você visualize este tutorial de introdução ao QGIS: bit.ly/13qVtdn.

Para criar uma nova camada vetorial, você deve utilizar o recurso apresentado pelo seguinte ícone, de nome "Camada do tipo shape": 

Logo após, o programa irá exibir a seguinte janela:



A camada será adicionada e a ela será atribuída uma cor aleatória. Observe que o SRC da camada já está definido para o nosso UTM.

Camada vetorial do RJ

Após a criação desses tutoriais, foram elaborados dois algoritmos em Python para o tratamento de cenas de satélite. O primeiro algoritmo faz a leitura de um pixel/célula específica de uma cena e retorna o deslocamento a partir da origem de x e de y, a referência geográfica desse pixel e o seu valor. Como exemplo e demonstração do algoritmo, foi baixado do inventário de cenas Landsat, disponível em: bit.ly/16KOOqD, a banda número oito do gomo UTM (Universal Transversa de Mercator) correspondente à região norte-fluminense (23S), a partir do sensor pancromático da família de satélites Landsat, com a maior resolução espectral e espacial. A imagem 1 a seguir mostra a respectiva banda inserida na área de trabalho do Quantum GIS (QGIS).

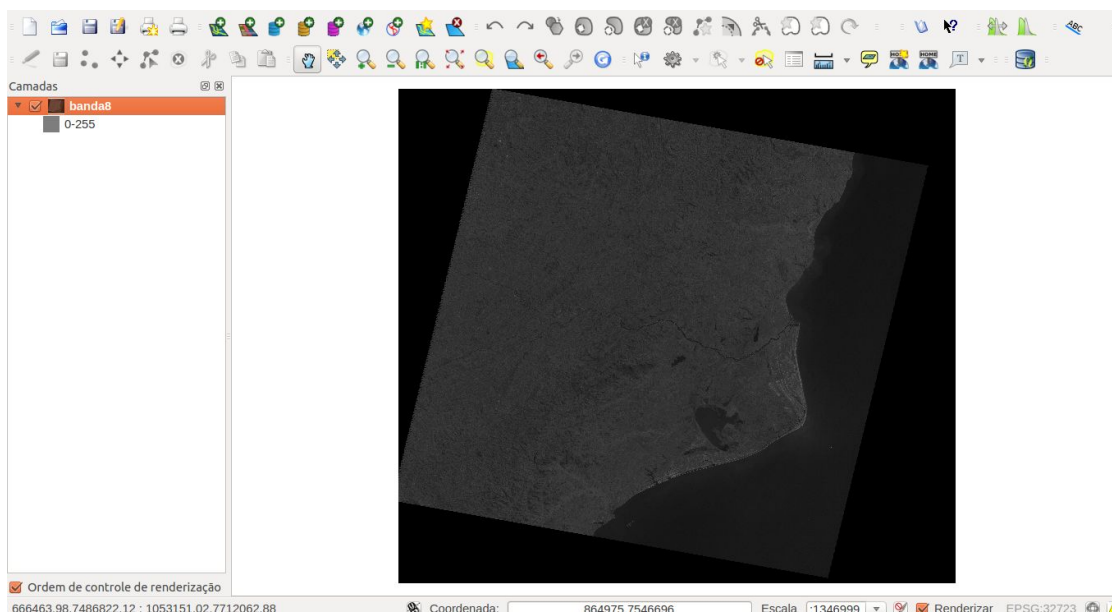


Imagem 1 - Banda 8 na área de trabalho do QGIS

A imagem 2 mostra uma ferramenta do QGIS que exibe a coordenada de um pixel específico e o seu valor ao clicar o cursor sobre ele. Neste exemplo, as coordenadas em x e y, respectivamente, são 841810 e 7611916, de acordo com o sistema de referência geográfica, o UTM zona 23S. A imagem também mostra que o valor do pixel nessa coordenada é de 58. Esse valor pertence à uma escala de 0 a 255 (base decimal 2 elevado a 8 bits, $2^8 = 256$ valores), onde 0 corresponde ao lado mais escuro (preto) dessa escala e 255 corresponde ao lado mais claro (branco).

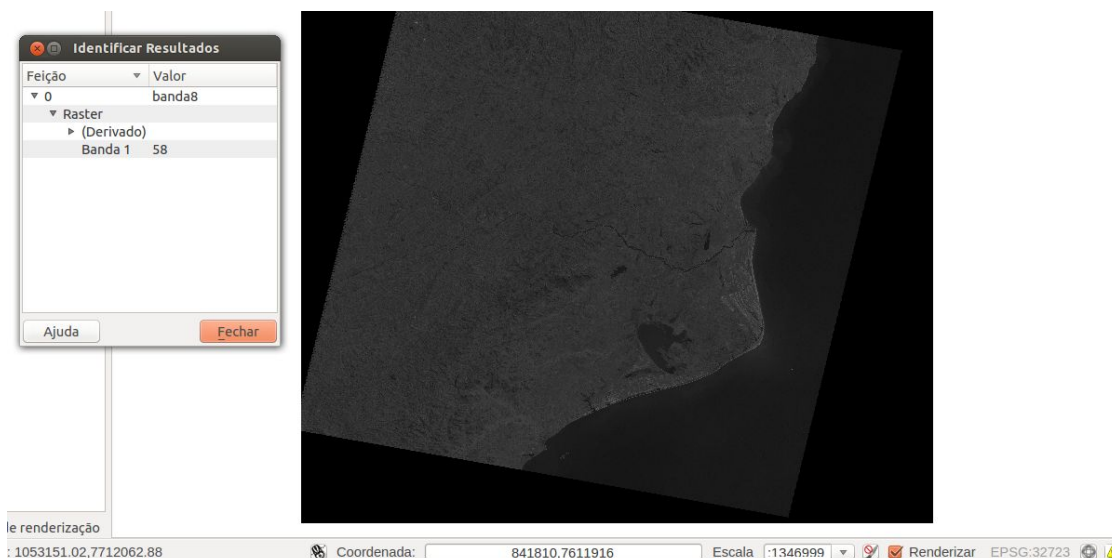


Imagem 2 - Ferramenta identificando o valor do pixel

Com o valor desse pixel é possível fazer o teste no algoritmo de leitura e verificar seu funcionamento, ao retornar o valor correto do pixel desejado. O algoritmo de leitura está descrito abaixo e já customizado para o exemplo trabalhado. Para implementação deste algoritmo, é necessária a biblioteca de código aberto GDAL, que suporta inúmeras linguagens de programação e é voltada para o suporte de dados geoespaciais em formato

raster (modelo matricial de imagem).

```
#coding:utf-8

import gdal, sys
from gdalconst import *

# registrar todos os drivers gdal, permitindo abrir cenas em qualquer formato
gdal.AllRegister()

# valor da coordenada do pixel desejado, em x e y
xValues = [841810.0]
yValues = [7611916.0]

# abrir a imagem
ds = gdal.Open("/home/neemiasvf/Dropbox/algebra-files/banda8.tif")
if ds is None:
    print 'Não foi possível abrir a imagem'
    sys.exit(1)

# obter tamanho da imagem
rows = ds.RasterYSize
cols = ds.RasterXSize

# resolução espectral
bands = ds.RasterCount

# obter informação da referência geográfica
transform = ds.GetGeoTransform()
xOrigin = transform[0]
yOrigin = transform[3]

#resolução espacial
pixelWidth = transform[1]
pixelHeight = transform[5]

# loop através das coordenadas
for i in range(1):

    # obter x e y
    x = xValues[i]
    y = yValues[i]

    # processar deslocamento de x e y a partir da origem
    xOffset = int((x - xOrigin) / pixelWidth)
    yOffset = int((y - yOrigin) / pixelHeight)

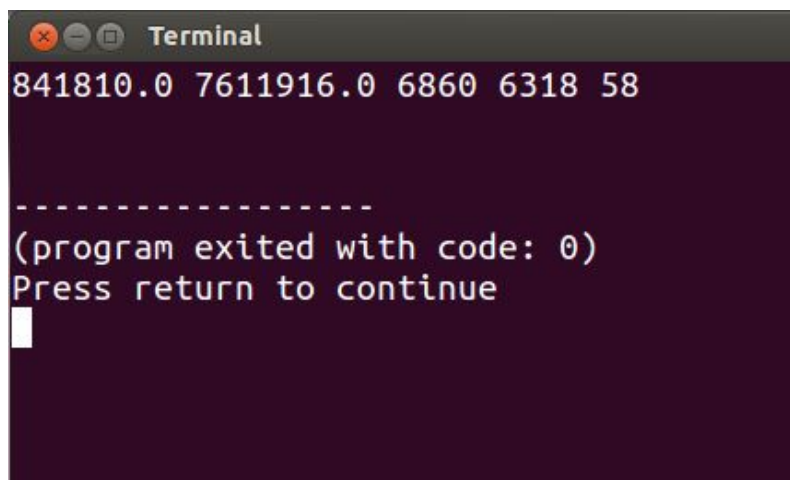
    # criar string para apresentação do resultado
    s = str(x) + ' ' + str(y) + ' ' + str(xOffset) + ' ' + str(yOffset) + ' '

    # loop através das bandas
```

```
for j in range(bands):
    band = ds.GetRasterBand(j+1)
    # ler dados e adicionar à string
    pixel = band.ReadAsArray(xOffset, yOffset, 1, 1)
    value = pixel[0,0]
    s = s + str(value) + ' '

# imprimir na tela o resultado
print s
```

Executando o algoritmo de leitura descrito acima, é retornada a string “841810.0 7611916.0 6860 6318 58”, onde cada valor corresponde respectivamente a coordenada x, coordenada y, deslocamento de x, deslocamento de y e valor do pixel. A imagem 3 abaixo mostra a execução do algoritmo no terminal do Ubuntu 13.04.



```
Terminal
841810.0 7611916.0 6860 6318 58

-----
(program exited with code: 0)
Press return to continue
```

Imagem 3 - Execução do algoritmo de leitura no terminal do Ubuntu

Com base na string retornada da execução do algoritmo, pode-se concluir que ele é funcional e atende ao objetivo proposto, a leitura dos dados de qualquer cena de satélite.

Com o algoritmo de leitura funcional, é hora do algoritmo de escrita ser testado quanto ao funcionamento. Esse algoritmo faz a leitura de cenas definidas pelo usuário, realiza um cálculo dos valores de cada pixel de cada cena a partir de uma equação também definida pelo usuário e escreve o valor de cada novo pixel resultante da equação em uma nova cena, salva em disco. O usuário pode definir qualquer cena e quantas desejar, assim como a equação de seu interesse, que irá manipular as cenas de tal forma que seja possível a visualização do fenômeno/característica desejada sobre a região que a cena representa, como a NDVI, por exemplo, citada anteriormente. Foi definida aleatoriamente para o exemplo abaixo a equação $(3 \times pb3^2) + (2 \times pb4) + (5 \times \sqrt{pb5})$, onde pb3, pb4 e pb5 são, respectivamente, o valor do pixel da banda 3, 4 e 5 do mesmo sensor do satélite Landsat referido anteriormente para a banda 8. Portanto, com base na equação, o algoritmo irá ler os pixels das bandas 3, 4 e 5, calcular a equação descrita e escrever o resultado em uma nova cena, chamada “banda345.tif” neste exemplo. Uma observação importante é a necessidade das cenas possuírem o mesmo tamanho (largura \times altura) para que o

algoritmo funcione, pois se elas não tiverem a quantidade exata de pixels, ele dará erro no momento do cálculo, uma vez que haverá mais ou menos pixels de uma banda/cena do que em outra,. Além da biblioteca GDAL, esse algoritmo necessita da biblioteca NumPy (computação científica) e utils (Garrard), obtidas através de terceiros.

```
#coding:utf-8
```

```
import os, sys, numpy, gdal, utils
from gdalconst import *
from math import sqrt
```

```
# registrar todos os drivers gdal, permitindo abrir cenas em qualquer formato
gdal.AllRegister()
```

```
# abrir a banda 3
```

```
inDs3 = gdal.Open("/home/neemiasvf/Dropbox/algebra-files/banda3.tif", GA_ReadOnly)
if inDs3 is None:
    print 'Não foi possível abrir a imagem'
    sys.exit(1)
```

```
# abrir a banda 4
```

```
inDs4 = gdal.Open("/home/neemiasvf/Dropbox/algebra-files/banda4.tif", GA_ReadOnly)
if inDs4 is None:
    print 'Não foi possível abrir a imagem'
    sys.exit(1)
```

```
# abrir a banda 5
```

```
inDs5 = gdal.Open("/home/neemiasvf/Dropbox/algebra-files/banda5.tif", GA_ReadOnly)
if inDs5 is None:
    print 'Não foi possível abrir a imagem'
    sys.exit(1)
```

```
# obter tamanho da banda 3 como parâmetro
```

```
rows = inDs3.RasterYSize
cols = inDs3.RasterXSize
bands = inDs3.RasterCount
```

```
# obter as bandas e o tamanho dos blocos de pixels
```

```
inBand3 = inDs3.GetRasterBand(1)
inBand4 = inDs4.GetRasterBand(1)
inBand5 = inDs5.GetRasterBand(1)
blockSizes = utils.GetBlockSize(inBand3)
xBlockSize = blockSizes[0]
yBlockSize = blockSizes[1]
```

```
# criar a imagem de saída
```

```
driver = inDs3.GetDriver()
outDs = driver.Create('/home/neemiasvf/Dropbox/algebra-files/banda345.tif', cols, rows, 1,
GDT_Float32)
if outDs is None:
    print 'Não foi possível criar banda345.tif'
```



```
sys.exit(1)
outBand = outDs.GetRasterBand(1)

# loop através das linhas
for i in range(0, rows, yBlockSize):
    if i + yBlockSize < rows:
        numRows = yBlockSize
    else:
        numRows = rows - i

    # loop através das colunas
    for j in range(0, cols, xBlockSize):
        if j + xBlockSize < cols:
            numCols = xBlockSize
        else:
            numCols = cols - j

    # ler dados de entrada
    data3 = inBand3.ReadAsArray(j, i, numCols, numRows).astype(numpy.float)
    data4 = inBand4.ReadAsArray(j, i, numCols, numRows).astype(numpy.float)
    data5 = inBand5.ReadAsArray(j, i, numCols, numRows).astype(numpy.float)

    # efetuar cálculos
    mask = numpy.greater((data3), 0)
    b345 = numpy.choose(mask, (-99, ((3*data3**2) + (2*data4) + (5*numpy.sqrt(data5)))))

    # escrever resultados
    outBand.WriteArray(b345, j, i)

# descarregar dados no disco, definir valores nulos e calcular estatísticas
outBand.FlushCache()
outBand.SetNoDataValue(-99)
stats = outBand.GetStatistics(0, 1)

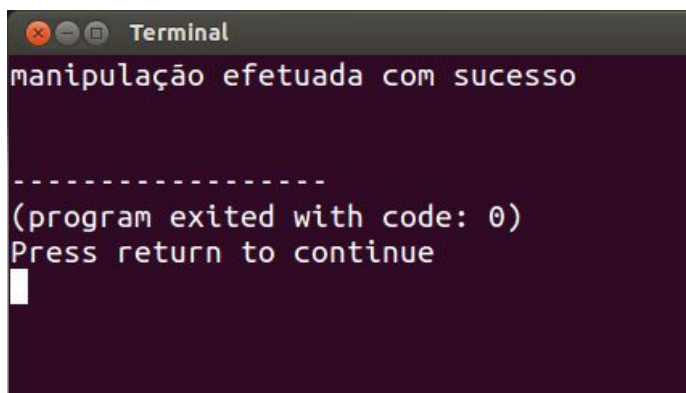
# georeferenciar a imagem e definir a projeção
outDs.SetGeoTransform(inDs3.GetGeoTransform())
outDs.SetProjection(inDs3.GetProjection())

# construir pirâmides
gdal.SetConfigOption('HFA_USE_RRD', 'YES')
outDs.BuildOverviews(overviewlist=[2,4,8,16,32,64,128])

inDs = None
outDs = None

print 'manipulação efetuada com sucesso'
```

A execução do algoritmo retornou a mensagem “manipulação efetuada com sucesso”, conforme a imagem 4 abaixo, indicando o funcionamento correto do algoritmo.

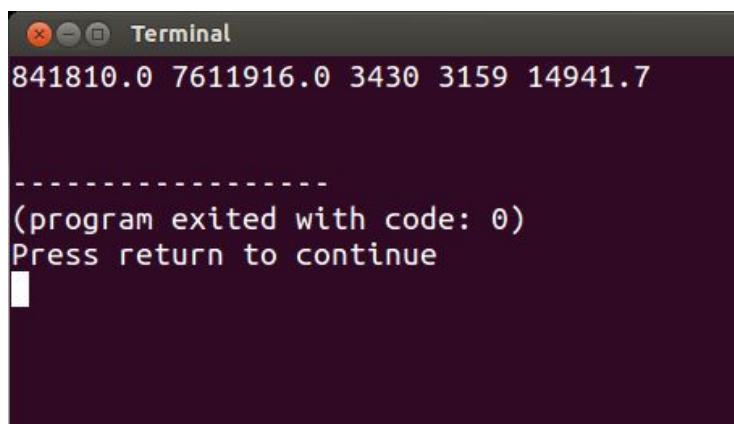


```
manipulação efetuada com sucesso

-----
(program exited with code: 0)
Press return to continue
```

Imagem 4 - Execução do algoritmo de escrita no terminal do Ubuntu

Para verificar se o cálculo foi efetuado corretamente, foram lidos os valores dos pixels de coordenadas x841810 e y7611916 das mesmas bandas 3, 4 e 5 com o algoritmo de leitura, retornando os valores 70, 96 e 99, respectivamente. Fazendo cálculo manual da equação definida no exemplo, o resultado do valor do pixel dessa coordenada deveria ser de 14941,7. Executando o algoritmo de leitura na cena “banda345.tif”, resultante do algoritmo de escrita, é retornada a string “841810.0 7611916.0 3430 3159 14941.7”, onde é possível observar que o cálculo foi feito corretamente, reafirmando o funcionamento do algoritmo. A imagem 5 mostra a apresentação dessa string.



```
841810.0 7611916.0 3430 3159 14941.7

-----
(program exited with code: 0)
Press return to continue
```

Imagem 5 - Execução do algoritmo de leitura na cena “banda345.tif”

8. CONCLUSÕES

Apesar do valor do pixel do exemplo anterior, assim como os restantes, não estar numa escala visível ao olho humano, o algoritmo atinge seu objetivo ao permitir ao usuário ler cenas de satélite, definir a equação desejada para a investigação de fenômenos e escrever a nova cena com o resultado do cálculo. Ainda assim, é possível redistribuir/equalizar os valores proporcionalmente em uma escala conveniente para inúmeras outras investigações, além da escala visível a olho nu. Também é possível utilizar a cena gerada em outros softwares que interpretam os valores e apresentam a informação relacionada. É recomendada a continuação deste projeto com a adição de novas funcionalidades como a de equalização dos valores dos pixels, além da implementação deste em plugin para softwares SIG, como o Quantum GIS, com interface intuitiva e dinâmica.

9. REFERÊNCIAS BIBLIOGRÁFICAS

- ESRI. **ESRI Shapefile Technical Description**. New York: Environmental Systems Research Institute, Inc., 1998. Disponível em: <bit.ly/Ypmgic>. Acesso em: 28 ago. 2013.
- FIGUEIREDO, D. **Conceitos Básicos de Sensoriamento Remoto**. Disponível em: <bit.ly/16OK32G>. Acesso em: 28 ago. 2013.
- FILHO, J. L. **Introdução a SIG - Sistemas de Informações Geográficas**. Universidade Federal do Rio Grande do Sul.
- FREITAS, Neemias Vitorino. **Referências Bibliográficas**. Google Drive: 2013. Disponível em: <bit.ly/19XiNnf>. Acesso em: 28 ago. 2013.
- FREITAS, Neemias Vitorino. **Introdução a geoprocessamento**. Google Drive: 2013. Disponível em: <bit.ly/18YKojn>. Acesso em: 28 ago. 2013.
- FREITAS, Neemias Vitorino. **Quantum GIS - Introdução**. Google Drive: 2013. Disponível em: <bit.ly/146AM2M>. Acesso em: 28 ago. 2013.
- FREITAS, Neemias Vitorino. **Quantum GIS - Camadas Vetoriais**. Google Drive: 2013. Disponível em: <bit.ly/1fheXT7>. Acesso em: 28 ago. 2013.
- GARRARD, Chris. **Geoprocessing with Python using Open Source GIS**. Utah: Utah State University, 2009. Disponível em: <bit.ly/17pCX81>. Acesso em: 28 ago. 2013.
- MENESES, P. R.; ALMEIDA, T. **Introdução ao processamento de imagens de sensoriamento remoto**. Brasília: CNPq, 2012. Disponível em: <bit.ly/16M0CsA>. Acesso em: 28 ago. 2013.
- PERNA, M. A. **O Sistema UTM**. Universidade do Estado do Rio de Janeiro. Disponível em: <bit.ly/19XnUUJ>. Acesso em: 28 ago. 2013.
- PINTO, Inês. **Introdução aos Sistemas de Informação Geográfica (SIG)**. Instituto de Investigação Científica Tropical, 2009. Disponível em: <bit.ly/YHINuY>. Acesso em: 28 ago. 2013.
- ROUSE, J. W.; HAAS, R. H.; SCHELL, J. A.; DEERING, D. W. **Monitoring vegetation systems in the Great Plains with ERTS**. In: ERTS-1 Symposium, 3, 10-14 December, Washington, DC. Proceedings. Washington, NASA SP-351, p. 309-317, 1973.
- Sustainability of Digital Formats Planning for Library of Congress Collections. **dBASE Table File Format (DBF)**. Disponível em: <1.usa.gov/1cehj5D>. Acesso em: 28 ago. 2013.
- Instituto Brasileiro de Geografia e Estatística. **Noções Básicas de Cartografia**. Disponível em: <<http://bit.ly/13JOPyL>>. Acesso em: 28 ago. 2013.
- Universidade Federal Fluminense. **Estudo Dirigido em SIG**. Disponível em: <bit.ly/Znv0Ko>. Acesso em: 28 ago. 2013.
- Instituto Nacional de Pesquisas Espaciais (INPE). Disponível em: <inpe.br/>. Acesso em: 28 ago. 2013.
- Instituto Nacional de Pesquisas Espaciais (INPE). **Divisão de Geração de Imagens (DGI)**. Disponível em: <dgi.inpe.br>. Acesso em: 28 ago. 2013.
- Instituto Nacional de Pesquisas Espaciais (INPE). **Divisão de Processamento de Imagens (DPI)**. Disponível em: <dpi.inpe.br>. Acesso em: 28 ago. 2013.
- Open Source Geospatial Foundation. **Quantum GIS**. Disponível em: <qgis.org>. Acesso em: 28 ago. 2013.
- Landsat Imagery. Disponível em: <bit.ly/16KOOqD>. Acesso em: 28 ago. 2013.

10. ATIVIDADES COMPLEMENTARES

- GARRARD, Chris. **Geoprocessing with Python using Open Source GIS**. Utah: Utah

State University, 2009. Disponível em: <bit.ly/17pCX81>. Acesso em: 28 ago. 2013.

Congresso Fluminense de Iniciação Científica e Tecnológica, V, 2013, Universidade Estadual do Norte Fluminense Darcy Ribeiro. A Ciência pela Água, Campos dos Goytacazes, 2013.

Fórum Internacional de Software Livre, 14, 2013, Pontifícia Universidade Católica do Rio Grande do Sul. A tecnologia que liberta, Porto Alegre, 2013.

11. AGRADECIMENTOS

Ao Instituto Federal de Educação, Ciência e Tecnologia Fluminense, por ter fomentado a pesquisa.

Campos dos Goytacazes, 28 de agosto de 2013.

Neemias Vitorino Freitas