Agrupamento de Escolas de Benfica

Escola Secundária José Gomes Ferreira

Grupo Disciplinar 510

Sistema Computacional para Geração Automática de Mapas Topográficos

Adrian Dias

Nº 1, T 12º 3ª

– Disciplina de Física –

2025 - 2026

Resumo

Este trabalho apresenta a modelação e visualização 3D da topografia da região metropolitana do Rio de Janeiro utilizando `Python` e a biblioteca `pygmt`. Descreve-se a implementação computacional de um mapa de elevação com perspectiva tridimensional, abrangendo uma área de aproximadamente 30x25 km. O modelo desenvolvido incorpora dados de relevo terrestre de alta resolução, técnicas de sombreamento e representação de curvas de nível, proporcionando uma análise detalhada da configuração topográfica da região.

1 Introdução

1.1 Enquadramento e Objectivos

O presente trabalho tem como objectivo principal a integração entre Python (com a biblioteca PyGMT) e LaTeX (via PythonTeX) para geração automática de cartas topográficas. A selecção desta abordagem justifica-se pela sua aplicação em contextos académicos e científicos, onde a precisão cartográfica e a automatização de processos são factores determinantes.

A metodologia adoptada integra conceitos de processamento de dados geoespaciais, visualização cartográfica e composição tipográfica, visando não apenas a reprodução

gráfica de cartas topográficas, mas também a compreensão dos princípios técnicos que governam a sua geração.

1.2 Fundamentos Técnicos do PyGMT e PythonTeX

1.2.1 PyGMT e Processamento de Dados Geoespaciais

O PyGMT constitui uma interface Python para o Generic Mapping Tools (GMT), oferecendo capacidades avançadas para manipulação de dados topográficos e geração de mapas de alta qualidade. A biblioteca suporta múltiplos formatos de dados, incluindo netCDF e GeoTIFF, e permite acesso a bases de dados públicas como o modelo global SRTM15+.

1.2.2 Integração com LaTeX via PythonTeX

O pacote PythonTeX permite a execução de código Python durante a compilação de documentos LaTeX, possibilitando a geração dinâmica de conteúdo. Esta abordagem garante reprodutibilidade e sincronização entre dados, cálculos e documentação.

2 Objectivos do Trabalho

- Integrar código Python com PyGMT num documento LaTeX utilizando PythonTeX
- Executar cálculos e gerar automaticamente uma carta topográfica hipsométrica da região do Rio de Janeiro
- Utilizar dados SRTM com resolução de 1 segundo de arco (30 metros)
- Documentar todo o procedimento, incluindo código, figuras e referências bibliográficas
- Implementar visualização 3D com perspectiva realista utilizando `grdview`

3 Metodologia

3.1 Abordagem Técnica

A metodologia consistiu numa abordagem integrada que combina processamento de dados geoespaciais com composição documental automatizada. O fluxo de trabalho compreendeu as seguintes etapas principais:

- Definição da região de estudo e parâmetros cartográficos
- Aquisição e processamento de dados topográficos via PyGMT
- Implementação de visualização 3D com configurações de perspectiva
- Integração do código Python no documento LaTeX via PythonTeX
- Processo de compilação em três fases para geração do documento final

3.2 Configuração do Ambiente

Para garantir reprodutibilidade, foram estabelecidas especificações técnicas precisas:

- **PyGMT**: Versão 0.16.0 para compatibilidade com dados SRTM15+
- **Python**: Versão 3.10 para suporte às bibliotecas necessárias
- **LaTeX**: Distribuição TeXLive com XeLaTeX e pacote PythonTeX
- **Dados**: Modelo SRTM15+ com resolução de 1 segundo de arco

4 Desenvolvimento

4.1 Definição da Área de Estudo

A região seleccionada para análise corresponde à área metropolitana do Rio de Janeiro, delimitada pelas seguintes coordenadas geográficas:

- **Longitude**: -43.35° a -43.05° (amplitude de 0.3°)
- **Latitude**: -23.05° a -22.83° (amplitude de 0.22°)

- **Área aproximada**: 30 × 25 km
- **Resolução espacial**: 1 segundo de arco (30 metros)

4.2 Implementação da Visualização 3D

4.2.1 Configuração da Perspectiva

A visualização tridimensional foi configurada com parâmetros optimizados para representação realista:

- **Azimute**: 135° (vista do sudeste)
- **Elevação**: 30° (ângulo de visualização balanceado)
- **Projecção**: Mercator (M18c) com escala de 18 cm
- **Altura vertical**: 0 a 3500 metros

4.2.2 Elementos Cartográficos

Foram implementados múltiplos elementos para enriquecimento cartográfico:

- **Coloração hipsométrica**: Paleta ”geo” para representação natural do relevo
- **Curvas de nível**: Intervalo de 100m com anotações a cada 500m
- **Iluminação e sombreamento**: Configuração +a45+nt0.8 para realce do relevo
- **Elementos costeiros**: Linha de costa com resolução fina e água semi-transparente

5 Resultados

5.1 Análise da Carta Topográfica Gerada

A implementação produziu uma carta topográfica 3D de alta qualidade (600 DPI) que demonstra capacidades avançadas de visualização geoespacial. A figura resultante apresenta as seguintes características técnicas:

- **Resolução espacial**: 1 segundo de arco (detalhamento de 30m)

- **Dimensões**: 18 cm de largura com proporções cartograficamente correctas
- **Elementos visuais**: Combinação de cores hipsométricas, curvas de nível e sombreamento
- **Anotações**: Escala de cores, título descritivo e elementos de orientação

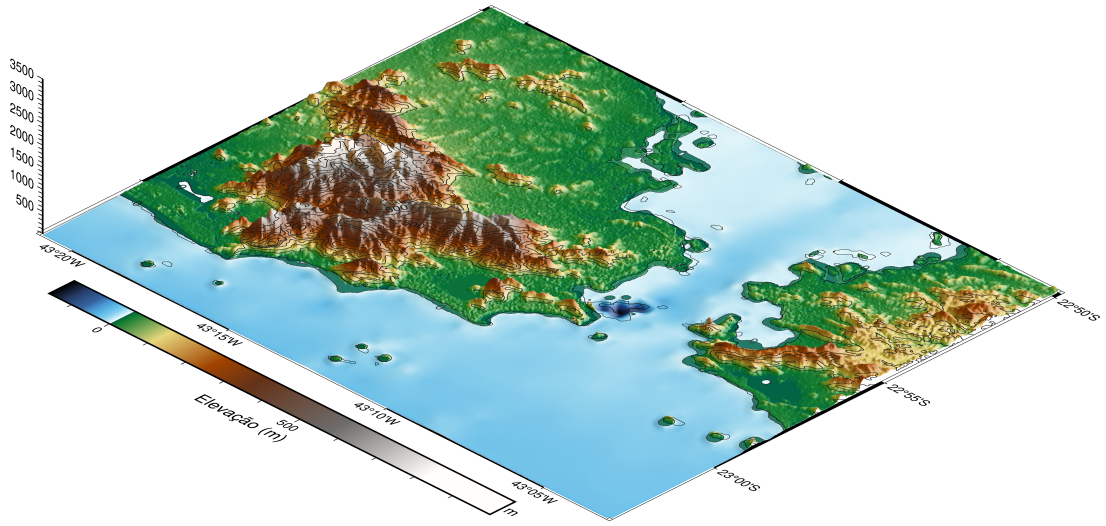

Figura 1: Carta topográfica 3D do Rio de Janeiro gerada automaticamente com PyGMT, mostrando o relevo da região com resolução de 30 metros.

5.2 Avaliação Técnica da Implementação

A integração PyGMT + PythonTeX + LaTeX mostrou-se robusta para geração automatizada de documentação científica. O processo de compilação em três fases garantiu:

- **Sincronização**: Actualização automática dos resultados com alterações no código
- **Qualidade gráfica**: Exportação vectorial com 600 DPI para impressão em escala A1
- **Reprodutibilidade**: Execução consistente em diferentes ambientes

6 Conclusão

6.1 Síntese das Principais Conquistas

O presente trabalho demonstra a viabilidade técnica da integração entre PyGMT e LaTeX via PythonTeX para produção automatizada de cartas topográficas. A abordagem desenvolvida permite:

- Geração dinâmica de visualizações geoespaciais directamente em documentos académicos
- Manutenção da qualidade cartográfica profissional em formatos vectoriais
- Automatização completa do fluxo desde dados brutos até documentação final

6.2 Limitações e Desafios Técnicos

Foram identificados alguns desafios durante o desenvolvimento:

- **Dependências complexas**: Requer configuração específica de múltiplas bibliotecas
- **Tempo de processamento**: Aquisição de dados SRTM pode demorar 30-60 segundos
- **Recursos computacionais**: Visualizações 3D exigem capacidade gráfica adequada

6.3 Perspectivas Futuras

Futuras evoluções do trabalho poderão incluir:

- Extensão para outras projecções cartográficas e escalas
- Implementação de animações temporais para análise de processos geomorfológicos
- Integração com bases de dados adicionais (geologia, hidrografia, etc.)
- Desenvolvimento de templates reutilizáveis para diferentes regiões geográficas

```
import pygmt

region = [-43.35, -43.05, -23.05, -22.83]  # ~30x25 km
print("Baixando grid SRTM 1s (~30m) - pode demorar 30-60s na primeira
    vez...")
grid = pygmt.datasets.load_earth_relief(
    resolution="01s",
    region=region,
    registration="gridline"
)
fig = pygmt.Figure()
perspective = [135, 30]  # Vista clássica do Rio (sudeste, 30° de
   elevação)
fig.grdview(
    grid=grid,
    region=region + [0, 3500],
    projection="M18c",
    perspective=perspective,
    frame=["a", "z500f100"],
    zsize="4c",
    surftype="i",
    shading="+a45+nt0.8",
    cmap="geo",
    contourpen="0.3p,gray30"
)
fig.grdcontour(
    grid=grid,
    levels=100,                    # a cada 100m
    annotation="500",              # anotar a cada 500m
    pen="0.4p,gray10",
    perspective=perspective
)
fig.coast(
    resolution="f",
    shorelines=True,
    water="lightblue@80",          # água semi-transparente
```

```
    borders="1/0.6p,gray",
    perspective=perspective
)
fig.colorbar(
    perspective=perspective,
    frame=["a500f100", "x+lElevação (m)", "y+lm"],
    position="JBR+o1c/1c+w14c/0.8c+h"
)
fig.text(
    text="Rio de Janeiro - Topografia 3D (30 m)",
    position="TC",
    font="18p,Helvetica-Bold,white",
    offset="0/0.8c",
    perspective=perspective,
    fill="black@90"  # fundo semi-transparente
)

fig.savefig("rio_3D_perfeito.png", dpi=600)
print("Mapa 3D salvo como 'rio_3D_perfeito.png' (600 DPI - pronto
   para impressão A1!)")

fig.show()
```

Referências