

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Engenharia de Computação - Turma OFT

Professor: Allbens Atman

Síntese de Fourier Complexa: Reconstrução de Imagen 2D

Aluno: Bernardo Vieira Rocha

Belo Horizonte, Minas Gerais
2025

1. Introdução:

Este relatório apresenta o desenvolvimento de uma aplicação prática da Transformada de Fourier Complexa para reconstrução de imagens em duas dimensões. O objetivo do trabalho é utilizar a decomposição de Fourier para representar o contorno de uma imagem através de uma soma de epicíclicos rotacionais, demonstrando visualmente o conceito matemático das séries de Fourier.

A imagem escolhida para reconstrução foi o logotipo do CEFET-MG, instituição na qual este trabalho foi desenvolvido. A técnica consiste em extrair os contornos da imagem e calcular os coeficientes complexos de Fourier que, quando somados, reproduzem o desenho original através de círculos rotacionando em diferentes frequências.

2. Fundamentação Teórica:

A Série de Fourier permite decompor qualquer função periódica em uma soma de funções senoidais. No caso de contornos em 2D, podemos representar as coordenadas (x, y) como funções complexas parametrizadas **no tempo t**, onde cada ponto é dado por $z(t) = x(t) + iy(t)$.

A síntese de Fourier complexa decompõe este contorno em uma série de epicíclicos, onde cada termo da série representa um círculo rotacionando em uma determinada frequência. Matematicamente, a função $z(t)$ pode ser expressa como:

$$z(t) = \sum c_n e^{in\omega t}$$

Onde c_n são os coeficientes complexos de Fourier calculados por:

$$c_n = \frac{1}{T} \int_0^T z(t) e^{-in\omega t} dt$$

Cada coeficiente c_n representa a amplitude e fase de um epicíclico específico. Frequências positivas rotacionam no sentido anti-horário, enquanto frequências negativas rotacionam no sentido horário.

3. Metodologia:

1. Processamento de Imagem

Inicialmente, a imagem do logo do CEFET-MG foi carregada e processada utilizando a biblioteca OpenCV. O processo incluiu conversão para escala de cinza, aplicação de filtro Gaussiano para suavização e limiarização binária pelo método de Otsu para obter uma imagem preto e branco com contornos bem definidos.

2. Extração de Contornos

Com a imagem binarizada, foram extraídos os contornos utilizando o algoritmo de detecção de contornos do OpenCV (`cv2.findContours`). Os contornos pequenos, considerados ruído, foram filtrados através de um limite mínimo de tamanho. Para o logo do CEFET-MG, foram identificados 2 contornos principais.

3. Conexão de Contornos

Os múltiplos contornos foram conectados de forma otimizada utilizando um algoritmo guloso de nearest neighbor. Este algoritmo seleciona o próximo contorno mais próximo e, quando necessário, reverte sua orientação para minimizar distâncias. Pontos de interpolação são adicionados entre contornos distantes para garantir uma transição suave.

4. Cálculo dos Coeficientes

Os coeficientes de Fourier foram calculados numericamente utilizando integração por quadratura (`scipy.integrate.quad_vec`). Para este trabalho, foram calculados 200 coeficientes ($N=200$), resultando em 401 coeficientes totais (incluindo frequências positivas e negativas e a componente DC).

5. Animação

A animação foi gerada utilizando `matplotlib.animation`, onde cada frame mostra os epicíclicos em sua posição correspondente ao tempo t . Os círculos são desenhados em azul, os vetores de conexão em verde, e o traço final do desenho em vermelho. A animação possui 150 frames e foi exportada em formato GIF.

4. Implementação:

The screenshot shows a Python code editor interface with the following details:

- Project Structure:** The left sidebar shows a project named "FOURIER-DRAW" containing "animations" (with "logoAnimada.gif"), "images" (with "logo_cefet.png"), "venv", and the script "fourier-draw-smooth.py".
- Code Editor:** The main area displays the source code for "fourier-draw-smooth.py". The code uses argparse to handle command-line arguments for input and output files, frame count, coefficient count, and contour settings.
- Terminal:** Below the code editor is a terminal window titled "TERMINAL". It shows the execution of the script with arguments "-i images/logo_cefet.png -o animations/logoAnimada.gif -f 500 -N 500". The terminal output indicates the process of loading the image, finding contours, calculating coefficients, and generating the final animation file.
- Status Bar:** At the bottom, there are tabs for "PROBLEMS", "OUTPUT", "TERMINAL", "GITLENS", and "GITLENS".

O código foi implementado em Python utilizando as seguintes bibliotecas principais:

- OpenCV (cv2): processamento de imagem e extração de contornos
 - NumPy: operações matemáticas e manipulação de arrays
 - SciPy: integração numérica para cálculo dos coeficientes
 - Matplotlib: visualização e geração da animação

Os parâmetros utilizados na execução foram:

- Imagem de entrada: logo_cefet.png
- Frames da animação: 500
- Coeficientes de Fourier (N): 500
- Tamanho mínimo de contorno: 10 pixels

5. Resultados:

1. Imagem Original:

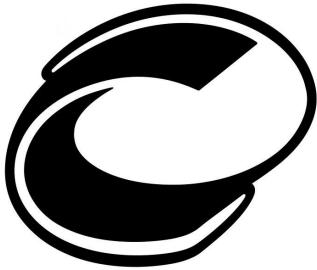


Figura 1: Logo do CEFET-MG
utilizado como entrada

2. Resultado da Animação:

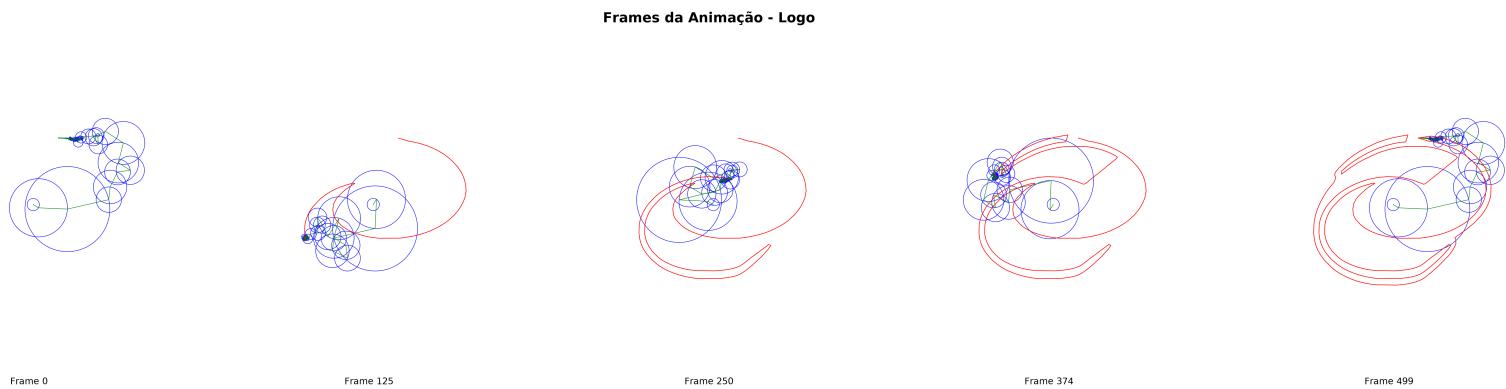


Figura 2: Frames da animação mostrando a progressão do desenho

Os círculos azuis representam os epicíclicos individuais, as linhas verdes conectam os centros dos círculos consecutivos, e a linha vermelha traça o caminho do ponto final, que gradualmente forma o logo completo do CEFET-MG.

A animação completa pode ser visualizada no arquivo GIF gerado (logoAnimada.gif). Com 500 coeficientes de Fourier, a reconstrução apresentou boa fidelidade ao contorno original, capturando as características principais do logo.

6. Análise de Resultados

A análise dos resultados obtidos permite observar alguns aspectos importantes sobre a síntese de Fourier complexa:

- **Convergência:** Com 200 coeficientes, a reconstrução apresentou qualidade satisfatória. Os primeiros coeficientes (baixas frequências) definem a forma geral do logo, enquanto coeficientes de ordem superior adicionam detalhes e refinamento ao contorno.
- **Conexão de contornos:** O algoritmo de conexão otimizada foi eficaz em unir os 2 contornos do logo, evitando saltos bruscos que causariam artefatos visuais indesejados na animação.
- **Suavização:** A aplicação do filtro Gaussiano na etapa de pré-processamento foi importante para eliminar ruídos de alta frequência, resultando em contornos mais limpos e uma animação mais fluida.
- **Performance computacional:** O tempo de execução foi adequado, com o cálculo dos coeficientes sendo a etapa mais custosa computacionalmente. A integração numérica converge bem com o limite de 100 iterações configurado.

