UnB

Relatório semestral de Iniciação Científica

Processamento e análise de dados de Controle Postural

Postural control data processing and analysis

Juliana de Souza Bispo[1]

[1]Universidade de Brasília

Resumo

Este projeto teve como base o estudo da trajetória do Centro de Massa Corporal e o Controle Postural no estudo de caso para pessoas com Síndrome de Down. O estudo sobre Controle Postural (CoP) é essencial para o entendimento da complexidade e variabilidade do equilíbrio corporal de uma pessoa, permitindo que atividades diárias sejam realizadas de forma segura. Movimentos do Centro de Massa Corporal (CMC) refletem o estado de equilíbrio, e ajustes contínuos são feitos pelo sistema neuromuscular para mantê-lo dentro da base de suporte e prevenir desequilíbrios e quedas.

Palavras-chave: *iniciação científica.*

I. Introdução

Proposta inicial do projeto era propor uma metodologia para a quantificação da condição de simetria da marcha humana, através da medição e avaliação da trajetória do centro de massa corporal utilizando um protótipo baseado em sensores inerciais. A metodologia proposta foi a seguinte:

Desenvolvimento do protótipo: O projeto eletrônico incluirá todas as etapas até chegar na obtenção do sinal medido: aquisição do sinal, processamento e transmissão. O desenvolvimento iniciará com simulações computacionais para testar a arquitetura projetada, seguida de testes em bancada, fabricação de placas de circuito impresso e finalmente testes de funcionamento.

Definição de protocolo de coleta de dados: O setup dos experimentos para obtenção dos resultados preliminares consistirá em um exercício de marcha feito em ambiente laboratorial onde será adquirido o sinal da trajetória do CoM proveniente do sistema desenvolvido, ao tempo que o MoCap medirá as trajetórias de marcadores colocados no voluntário para obter

o sinal do CoM estimado com o sistema de referência.

Coleta e análise de dados: Os experimentos serão realizados no Laboratório de Análise do Movimento Humano e de Processamento de Sinais da Faculdade de Ceilândia da UnB, seguindo o protocolo definido na etapa anterior. Após a execução do experimentos, os dados obtidos com os dois sistemas serão processados para obter as duas estimativas de CoM. Na sequência, o cálculo de nível de simetria utilizando o método detalhado será executado com o propósito de fazer uma comparação dos resultados obtidos com o sistema proposto e o de referência.

II. Metodologia

O trabalho nesses primeiros 6 meses foram divididos em 3 principais etapas:

Primeira etapa: Maio

No primeiro mês foi feito um estudo sobre o parâmetro de Centro de Massa Corporal e as possíveis formas de calcular a sua trajetória para o estudo da simetria da marcha humana.

Segunda etapa: Junho - Agosto

Nos meses de junho até agosto foi feita a implementação de códigos em Python para a analise de dados já extraídos anteriormente. Os dados foram coletados por um grupo de 15 pacientes atípicos com síndrome de Down.

Primeiramente foi criado um código simples para plotar os dados salvos em um arquivo txt, sem filtrar ou transformar os dados, os dados foram plotados em relação ao tempo em segundos e divididos de acordo com a natureza da medição.

Em seguida, o novo código de Python criado foi escrito com o objetivo de desenhar em um gráfico a trajetória do centro de massa corporal durante a medição, tanto no 3D quando no 2D, os dados usados foram obtidos através do acelerômetro e convertidos de aceleração para deslocamento. A partir dos resultados dos gráficos, foi observado que as medições foram feitas enquanto os pacientes estavam imoveis apresentando algumas oscilações consequentes do Controle Postural que o paciente tem.

Terceira etapa: Setembro - Outubro

No meses de setembro e outubro, o foco do projeto mudou para o estudo de uma nova dissertação sobre o Controle Postural e os métodos de cálculo de Multiscale Entropy e Sample Entropy que podem ser usados no estudo de CoP. No primeiro momento foi feita a revisão de alguns artigos sobre o tema e a construção de uma tabela comparativa entre

a bibliografia usada de base. Em seguida foi feito um código de Python para aplicar os cálculos, de Multiscale Entropy e Sample Entropy, e analisar os dados anteriores com um foco no Controle Postural de cada paciente.

III. Resultados

Primeira etapa

Abaixo segue as imagens dos plots obtidos a partir das medições feitas anteriormente:

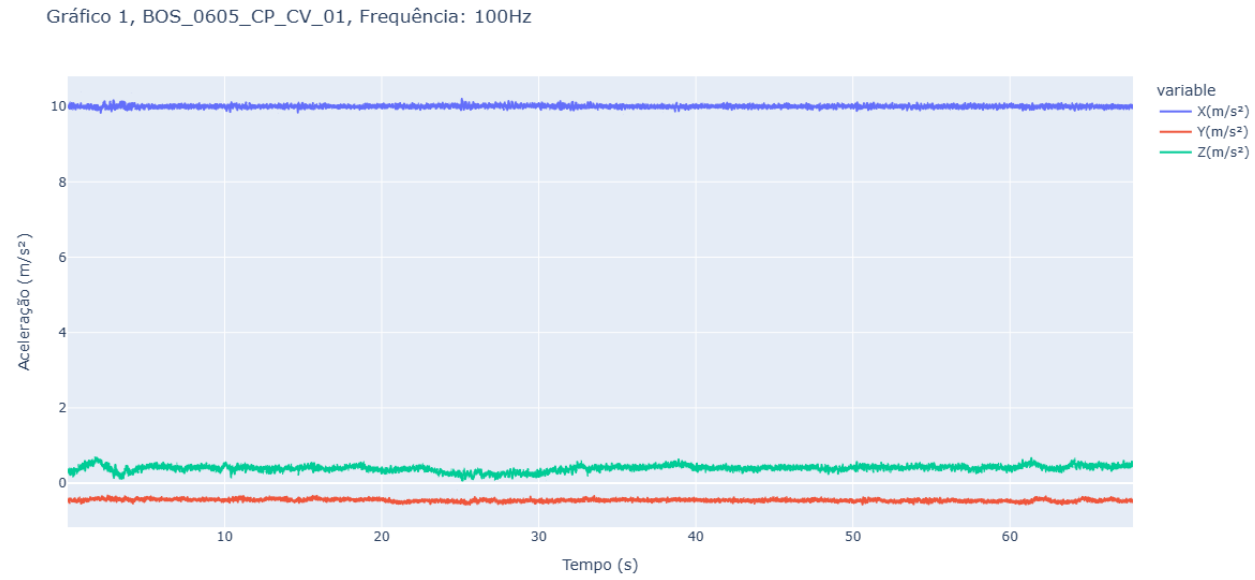

Figura 1: *Dados plotados obtidos do acelerômetro, nos 3 eixos*

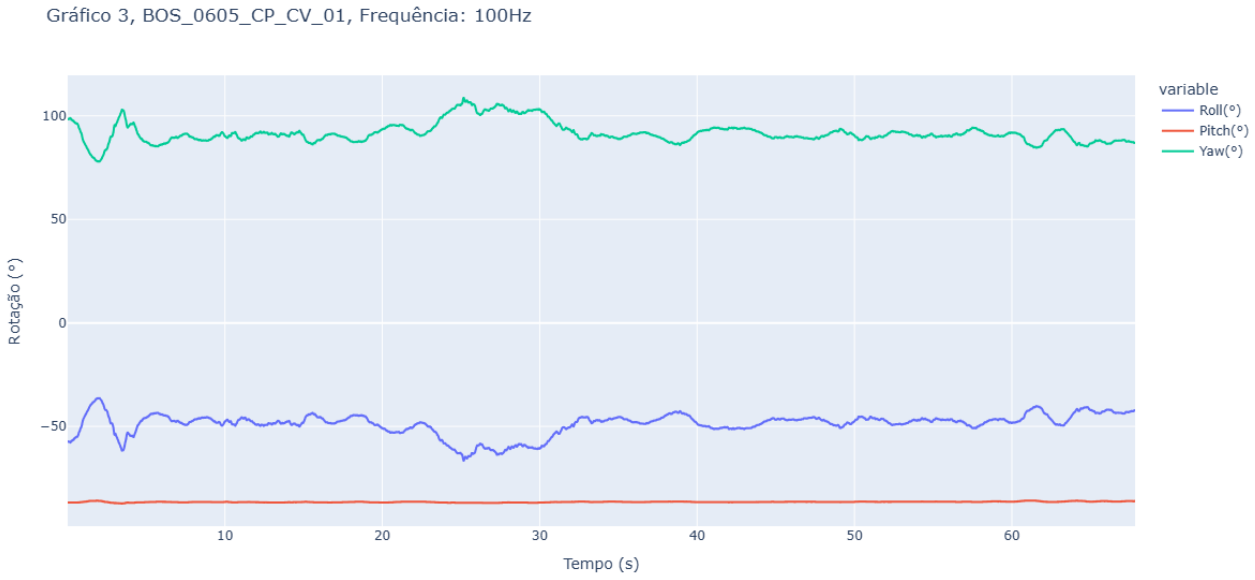

Figura 2: *Dados plotados obtidos da velocidade angular, nos 3 eixos*

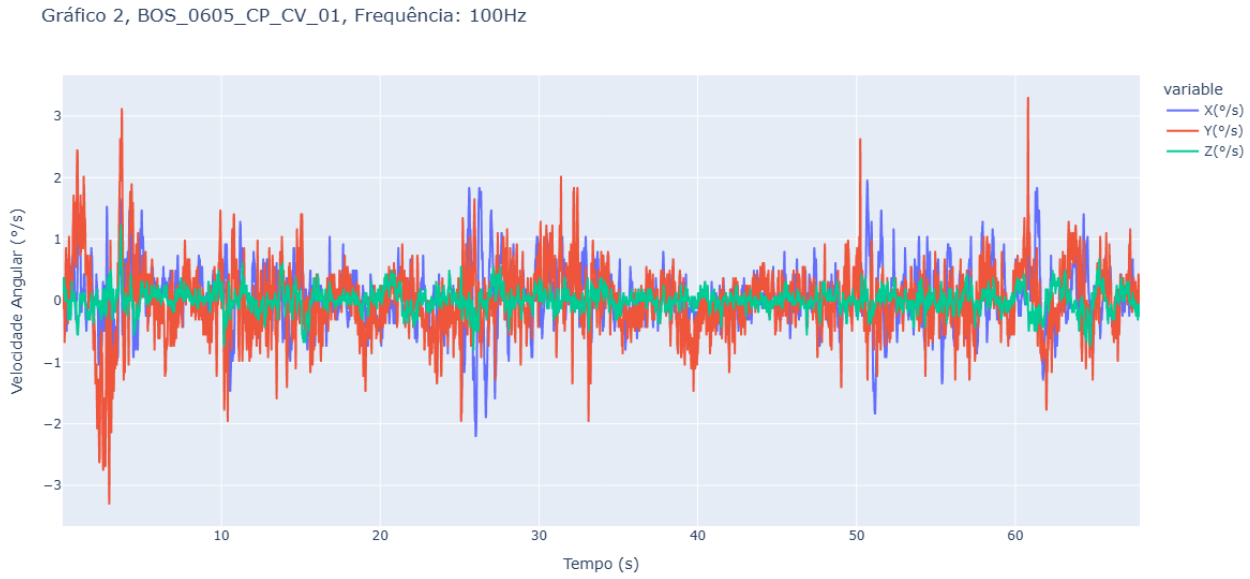

Figura 3: *Dados plotados obtidos da rotação, nos 3 eixos*

Segunda etapa

Abaixo segue os plots obtidos através da decomposição da aceleração, os plots no 3D usando linha e pontilhado.

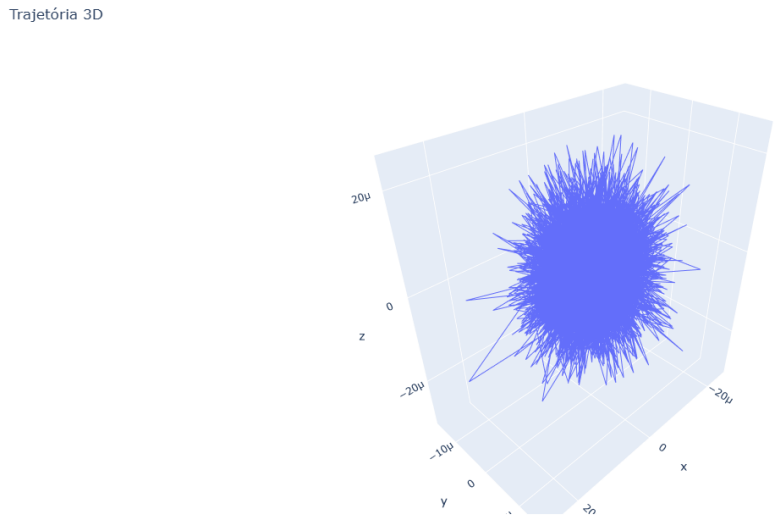

Figura 4: *Dados plotados obtidos do deslocamento no 3d*

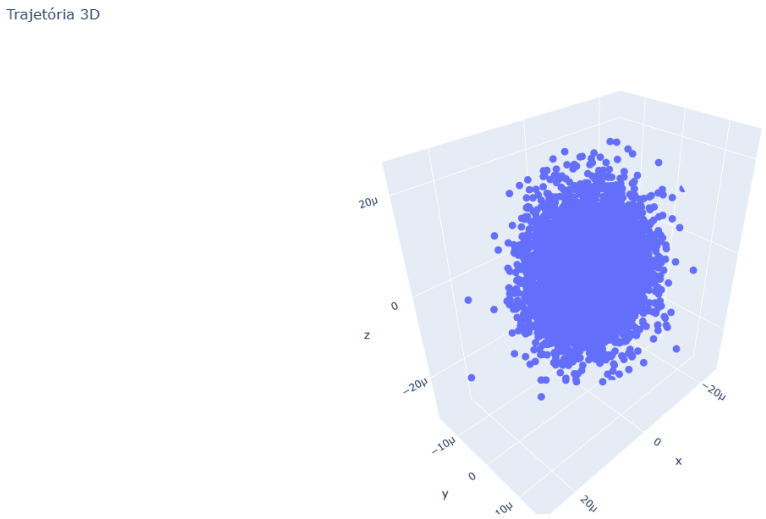

Figura 5: *Dados plotados usando gráfico de dispersão obtidos do deslocamento no 3d*

Terceira etapa

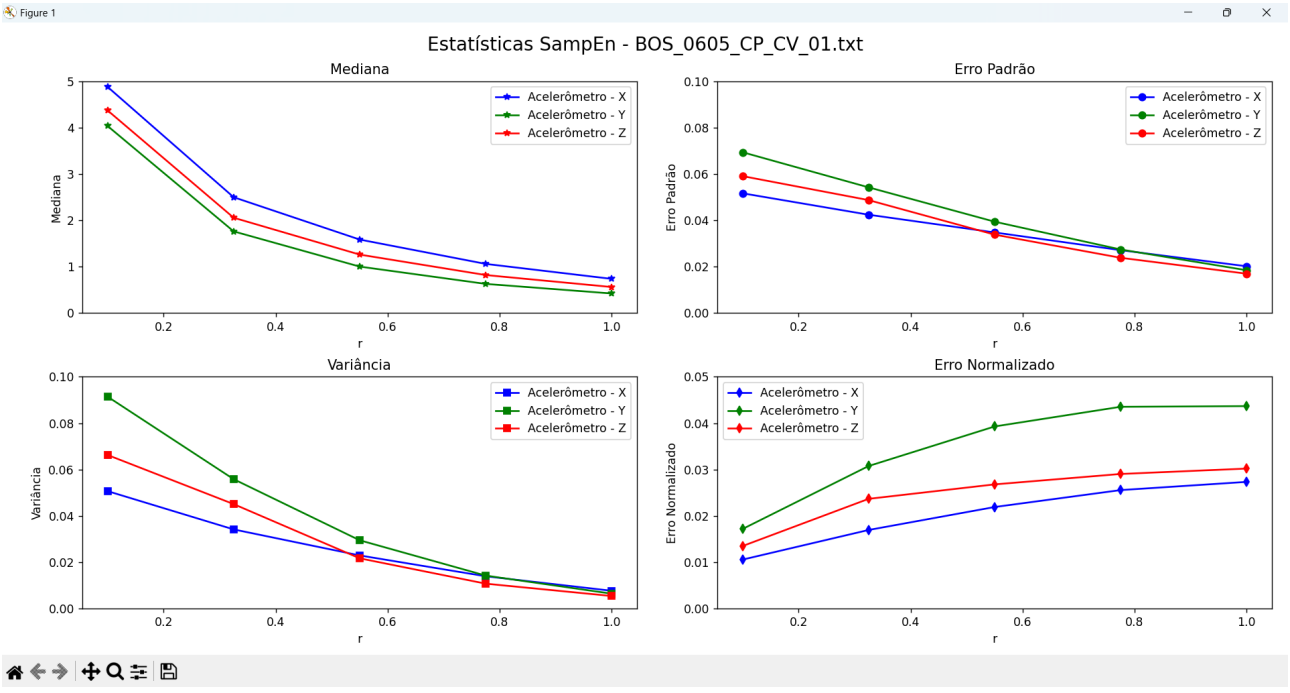

Figura 6: *Dados plotados obtidos do acelerômetro, nos 3 eixos*

```
    Análise dos Gráficos:
    1. **Mediana SampEn**:
       Este gráfico mostra a mediana da entropia amostral
(SampEn) para os dados.
       O valor de 'r' varia entre 0,1 e 1, representando
diferentes tolerâncias.
       A mediana indica a complexidade global do padrão de
movimento.
       Valores mais altos de mediana sugerem maior
variabilidade nos dados.

    2. **Erro Padrão SampEn**:
       O erro padrão da mediana indica a precisão dos
cálculos de SampEn.
       Quanto menor o erro, mais confiável é a estimativa da
mediana.

    3. **Variância SampEn**:
       A variância mede a dispersão dos valores de SampEn.
```

```
       Valores mais altos de mediana sugerem maior
variabilidade nos dados.

    2. **Erro Padrão SampEn**:
       O erro padrão da mediana indica a precisão dos
cálculos de SampEn.
       Quanto menor o erro, mais confiável é a estimativa da
mediana.

    3. **Variância SampEn**:
       A variância mede a dispersão dos valores de SampEn.
       Uma maior variância indica maior irregularidade ou
imprevisibilidade nos dados.

    4. **Erro Normalizado SampEn**:
       O erro normalizado ajusta a precisão da mediana
levando em conta sua magnitude.
       Um erro menor significa que a mediana é bem estimada
e confiável.
```

Figura 7: *Quadro com explicação de como analisar cada gráfico*

Referências

D. I. C. Damacena, "Sistema de medição angular em juntas articuladas baseado em imu," 2023. 8

G. L. M. de Oliveira, "Estimativa da trajetória do centro de massa corporal utilizando blazepose," 2021. 8

L. P. Silva, "Controle postural em crianças com transtorno do espectro autista: Influência de diferentes demandas na regularidade do controle postural," Master's thesis, Universidade Federal de Minas Gerais, 2024. 8

R. de Carvalho, "Efeito da fadiga neuromuscular sobre a regularidade do movimento durante a corrida," Master's thesis, Universidade Federal de Minas Gerais, 2020. 8

M. Costa, A. L. Goldberger, and C.-K. Peng, "Multiscale entropy analysis of complex physiologic time series," *Physica A*, 2002. 8

M. Costa, A. L. Goldberger, and C.-K. Peng, "Multiscale entropy analysis (mse)," *PHYSICAL REVIEW LETTERS*, 2002. 8

M. Duarte and S. M. S. F. Freitas, "Revision of posturography based on force plate for balance evaluation," *Revista Brasileira de Fisioterapia*, 2010. 8

B. J. Gow, "Multiscale entropy analysis of center-of-pressure dynamics in human postural control: Methodological considerations," *MDPI Journals*, 2015. 8

C. D. Napoli, "Postural complexity during listening in young and middle-aged adults," *MDPI Journals*, 2022. 8

T. M. Pires, "Quantificação da complexidade do ritmo cardíaco usando o método da multiscale entropy," Master's thesis, Universidade Nova de Lisboa, 2011. 8

S. Ramdani, "On the use of sample entropy to analyze human postural sway data," *Medical Engineering Physics*, 2009. 8

J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology*, 2000. 8

N. Stergiou and L. M. Decker, "Human movement variability, nonlinear dynamics, and pathology: Is there a connection?," *Human Movement Science*, 2011. 8

S. de Oliveira Veronez, "The use of nonlinear analysis in understanding postural control: A scoping review," *Human Movement Science*, 2024. 8

Inuritdino, "Multiscaleentropy repository." <https://github.com/inuritdino/MultiScaleEntropy>, 2023. Accessed: 2024-11-07. 8
[2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]

IV. Apêndice A

Código em Python para plotar 3 gráficos com os dados brutos

```
import pandas as pd
import plotly.express as px
import numpy as np
import os
from pathlib import Path

# Função para plotar a matriz em gráficos diferentes com zoom interativo e salvar os gráficos
def plotar_matriz_interativa(tabela, output_dir, nome_arquivo):
    # Define as colunas para cada gráfico
    colunas = [
        [1, 2, 3],  # Gráfico 1: Colunas 1, 2 e 3
        [4, 5, 6],  # Gráfico 2: Colunas 4, 5 e 6
        [7, 8, 9]   # Gráfico 3: Colunas 7, 8 e 9
    ]

    nomes_linhas = [
        ['X(m/sš)', 'Y(m/sš)', 'Z(m/sš)'],  # Nomes para o gráfico 1
        ['X(ř/s)', 'Y(ř/s)', 'Z(ř/s)'],     # Nomes para o gráfico 2
        ['Roll(ř)', 'Pitch(ř)', 'Yaw(ř)']   # Nomes para o gráfico 3
    ]

    y_labels = [
        'Aceleração (m/sš)',          # Nome do eixo y para o gráfico 1
        'Velocidade Angular (ř/s)',  # Nome do eixo y para o gráfico 2
        'Rotação (ř)'                 # Nome do eixo y para o gráfico 3
    ]

    nome_arquivo_sem_extensao = os.path.splitext(os.path.basename(nome_arquivo))[0]

    # Itera sobre cada conjunto de colunas e nomes de linhas para criar os gráficos
    for i, (cols, nomes) in enumerate(zip(colunas, nomes_linhas)):
        # Verifica se a coluna existe no DataFrame
```

```
        if all(tabela.columns[col] in tabela.columns for col in cols):
            # Cria um gráfico de linha para as colunas especificadas
            fig = px.line(tabela, x=tabela.columns[0], y=[tabela.columns[col] for col in cols],
                          labels={'x': 'Tempo(s)', 'value': y_labels[i]},
                          title=f'Gráfico {i + 1}, {nome_arquivo_sem_extensao}, Frequência: 100Hz')

            # Define o nome da linha no gráfico
            for j, nome in enumerate(nomes):
                fig.data[j].name = nome  # Define o nome da linha no gráfico

            # Atualiza o layout do gráfico para garantir que o eixo y fique normal sem transformações
            fig.update_layout(
                xaxis_title='Tempo (s)',
                yaxis_title=y_labels[i],
                yaxis=dict(type='linear'),  # Certifica-se de que o eixo y é linear
                width=1200,
                height=600
            )

            # Salva o gráfico na pasta de saída
            output_file = os.path.join(output_dir, f'Grafico_{i + 1}.html')
            fig.write_html(output_file)
            print(f'Gráfico {i + 1} salvo em {output_file}')
        else:
            print(f"Colunas {cols} não encontradas no DataFrame.")

# Função principal para ler todos os arquivos e gerar os gráficos
def processar_arquivos_na_pasta(pasta):
    # Obtém a lista de todos os arquivos .txt na pasta
    arquivos = [f for f in os.listdir(pasta) if f.endswith('.txt')]

    for arquivo in arquivos:
        caminho_arquivo = os.path.join(pasta, arquivo)
        # Lê o arquivo .txt, usa o tab como quebra, ignora as 14
```

```
            linhas de cabeçalho e identifica a ',' como separaçao de
                decimais
67          tabela = pd.read_csv(caminho_arquivo, sep="\t", header=13,
                decimal=',')
68
69          # Cria uma nova pasta para salvar os gráficos, com o nome
                do arquivo (sem extensão)
70          nome_pasta_saida = os.path.join(pasta, Path(arquivo).stem)
71          os.makedirs(nome_pasta_saida, exist_ok=True)
72
73          # Plota e salva os gráficos
74          plotar_matriz_interativa(tabela, nome_pasta_saida, arquivo)
75
76  # Define o caminho da pasta onde estão os arquivos
77  pasta_dos_arquivos = r"C:\Users\julia\controle-postural"
78
79  # Chama a função para processar todos os arquivos na pasta
80  processar_arquivos_na_pasta(pasta_dos_arquivos)
```

V. Apêndice B

Código em Python para plotar o deslocamento em 3D pelo tempo

```
1   import numpy as np
2   import pandas as pd
3   import plotly.express as px
4
5   def carregar_dados(caminho_arquivo, skip_rows=15, sep="\t", decimal
        =','):
6       try:
7           return pd.read_csv(caminho_arquivo, skiprows=skip_rows, sep
                =sep, decimal=decimal)
8       except FileNotFoundError:
9           print(f"Arquivo não encontrado: {caminho_arquivo}")
10          return None
11
12  def calcular_velocidade(aceleracao, delta_t):
13      velocidade = np.diff(aceleracao) * delta_t
14      velocidade = np.insert(velocidade, 0, 0)  # Adiciona um zero no
            início para manter o mesmo tamanho
15      return velocidade
16
17  def calcular_posicao(velocidade, delta_t):
18      posicao = np.diff(velocidade) * delta_t
```

```
19      posicao = np.insert(posicao, 0, 0)  # Adiciona um zero no
            início para manter o mesmo tamanho
20      return posicao
21
22  def plotar_trajetoria_3d(df):
23      fig = px.line_3d(df, x='x', y='y', z='z', title="Trajetória␣3D"
            )
24      fig.update_traces(marker=dict(size=5))
25      fig.update_layout(scene=dict(xaxis=dict(range=[min(df['x']),
            max(df['x'])]),
26                                   yaxis=dict(range=[min(df['y']),
                                         max(df['y'])]),
27                                   zaxis=dict(range=[min(df['z']),
                                         max(df['z'])])))
28      fig.show()
29
30  def plotar_grafico_temporal(tempo, eixo, nome_eixo):
31      fig = px.line(x=tempo, y=eixo, labels={'x': 'Tempo␣(s)', 'y':
            nome_eixo}, title=f"{nome_eixo}␣vs␣Tempo")
32      fig.update_yaxes(range=[min(eixo), max(eixo)])  # Ajuste manual
            do range do eixo Y
33      fig.show()
34
35  # Parâmetros
36  delta_t = 0.010  # Intervalo de tempo em segundos
37
38  # Carregar os dados
39  dados = carregar_dados(r"C:\Users\julia\controle-postural\
        BOS_0605_CP_CV_01.txt")
40
41  if dados is not None:
42      # Extraindo as colunas
43      tempo = dados.iloc[:, 0].values
44      ax = dados.iloc[:, 1].values
45      ay = dados.iloc[:, 2].values
46      az = dados.iloc[:, 3].values
47
48      # Calculando a velocidade
49      vx = calcular_velocidade(ax, delta_t)
50      vy = calcular_velocidade(ay, delta_t)
51      vz = calcular_velocidade(az, delta_t)
52
53      # Calculando a posição
54      x = calcular_posicao(vx, delta_t)
```

```
55      y = calcular_posicao(vy, delta_t)
56      z = calcular_posicao(vz, delta_t)
57
58      # Criando o dataframe para plotar
59      df = pd.DataFrame({'x': x, 'y': y, 'z': z})
60
61      # Plotando o gráfico 3D
62      plotar_trajetoria_3d(df)
63
64      # Plotando os gráficos temporais
65      plotar_grafico_temporal(tempo, x, 'Posição X')
66      plotar_grafico_temporal(tempo, y, 'Posição Y')
67      plotar_grafico_temporal(tempo, z, 'Posição Z')
68
69      # Impressões para verificação
70      print(ax[:10], ay[:10], az[:10])  # Mostra os primeiros 10
            valores
71      print(vx[:10], vy[:10], vz[:10])
72      print(x[:10], y[:10], z[:10])
73      print(df.head(10))  # Verifica os primeiros valores no
            DataFrame
```

VI. Apêndice C

Código em Python para plotar os graficos de analise do SampEn

```
1  import numpy as np  # Biblioteca para operações matemáticas e
       arrays
2  import pandas as pd  # Biblioteca para manipulação de dados em
       DataFrames
3  import matplotlib.pyplot as plt  # Biblioteca para criação de
       gráficos
4  import os  # Biblioteca para manipulação de caminhos de arquivos
5  import tkinter as tk  # Biblioteca para criar interfaces gráficas
6  from tkinter import scrolledtext  # Para criar uma caixa de texto
       rolável
7  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg  #
       Para embutir gráficos no Tkinter
8
9  # Função para carregar os dados de um arquivo
10 def carregar_dados(caminho_arquivo, skip_rows=15, sep="\t", decimal
       =','):
11     try:
```

```
12          return pd.read_csv(caminho_arquivo, skiprows=skip_rows, sep
                =sep, decimal=decimal)
13      except FileNotFoundError:
14          print(f"Arquivo não encontrado: {caminho_arquivo}")
15          return None
16
17  # Função para calcular a entropia amostral (SampEn)
18  def sample_entropy(time_series, m, r):
19      n = len(time_series)
20
21      def _phi(m):
22          x = np.array([time_series[i:i + m] for i in range(n - m +
                1)])
23          C = np.sum(np.max(np.abs(x[:, None] - x[None, :]), axis=2)
                <= r, axis=0) - 1
24          return np.sum(C) / (n - m + 1) / (n - m)
25
26      return -np.log(_phi(m + 1) / _phi(m))
27
28  # Função para calcular a entropia multiescala (MSE)
29  def multiscale_entropy(data, scale_max, m=2, r=0.2):
30      def coarse_graining(data, scale):
31          n = len(data)
32          b = n // scale
33          return np.mean(data[:b * scale].reshape((b, scale)), axis
                =1)
34
35      entropies = []
36      for scale in range(1, scale_max + 1):
37          coarse_data = coarse_graining(data, scale)
38          sampen = sample_entropy(coarse_data, m=m, r=r * np.std(
                coarse_data))
39          entropies.append(sampen)
40
41      return entropies
42
43  # Função para calcular estatísticas das entropias SampEn XY
44  def calcular_estatisticas_sampen(mse_x, mse_y):
45      sampen_xy = np.array(mse_x) + np.array(mse_y)
46
47      mediana_sampen_xy = np.median(sampen_xy)
48      erro_padrao_sampen_xy = np.std(sampen_xy, ddof=1) / np.sqrt(len
            (sampen_xy))
49      variancia_sampen_xy = np.var(sampen_xy)
```

```
    erro_normalizado_sampen_xy = erro_padrao_sampen_xy /
       mediana_sampen_xy

    return (mediana_sampen_xy, erro_padrao_sampen_xy,
       variancia_sampen_xy, erro_normalizado_sampen_xy)

# Função para plotar as estatísticas para os três eixos no mesmo
   gráfico
def plotar_estatisticas_combinadas(r_vals, nome_arquivo,
   estatisticas):
    fig, axes = plt.subplots(2, 2, figsize=(12, 8))

    # Usar o nome do arquivo como título do gráfico
    titulo_plot = f'Estatísticas␣SampEn␣-␣{os.path.basename(
       nome_arquivo)}'
    fig.suptitle(titulo_plot, fontsize=16)

    cores = {'X': 'b', 'Y': 'g', 'Z': 'r'}  # Cores para os eixos

    # Iterar sobre os gráficos e plotar para cada eixo (X, Y e Z)
    for eixo, (mediana_vals, erro_padrao_vals, variancia_vals,
       erro_normalizado_vals) in estatisticas.items():
        axes[0, 0].plot(r_vals, mediana_vals, marker='*', color=
           cores[eixo], label=f'Acelerômetro␣-␣{eixo}')
        axes[0, 1].plot(r_vals, erro_padrao_vals, marker='o', color
           =cores[eixo], label=f'Acelerômetro␣-␣{eixo}')
        axes[1, 0].plot(r_vals, variancia_vals, marker='s', color=
           cores[eixo], label=f'Acelerômetro␣-␣{eixo}')
        axes[1, 1].plot(r_vals, erro_normalizado_vals, marker='d',
           color=cores[eixo], label=f'Acelerômetro␣-␣{eixo}')

    # Configurar títulos, legendas e limites dos eixos Y
    axes[0, 0].set_title('Mediana')
    axes[0, 0].set_xlabel('r')
    axes[0, 0].set_ylabel('Mediana')
    axes[0, 0].set_ylim(0, 5)
    axes[0, 0].legend()

    axes[0, 1].set_title('Erro␣Padrão')
    axes[0, 1].set_xlabel('r')
    axes[0, 1].set_ylabel('Erro␣Padrão')
    axes[0, 1].set_ylim(0, 0.1)
    axes[0, 1].legend()

```

```
84      axes[1, 0].set_title('Variância')
85      axes[1, 0].set_xlabel('r')
86      axes[1, 0].set_ylabel('Variância')
87      axes[1, 0].set_ylim(0, 0.1)
88      axes[1, 0].legend()
89
90      axes[1, 1].set_title('Erro Normalizado')
91      axes[1, 1].set_xlabel('r')
92      axes[1, 1].set_ylabel('Erro Normalizado')
93      axes[1, 1].set_ylim(0, 0.05)
94      axes[1, 1].legend()
95
96      plt.tight_layout()
97      return fig
98
99  # Função para exibir a explicação e o gráfico na interface Tkinter
100 def abrir_interface():
101     # Criar a janela principal
102     root = tk.Tk()
103     root.title("Análise de Entropia Amostral")
104
105     # Criar um layout de grid com duas colunas
106     root.grid_rowconfigure(0, weight=1)
107     root.grid_columnconfigure(0, weight=1)
108     root.grid_columnconfigure(1, weight=1)
109
110     # Criar o widget de texto rolável para a explicação
111     explicacao_texto = scrolledtext.ScrolledText(root, width=60,
          height=20, wrap=tk.WORD)
112     explicacao_texto.grid(row=0, column=1, padx=10, pady=10)
113
114     # Adicionar a explicação no widget de texto
115     explicacao_texto.insert(tk.INSERT, """
116     Análise dos Gráficos:
117     1. **Mediana SampEn**:
118        Este gráfico mostra a mediana da entropia amostral (SampEn)
             para os dados.
119        O valor de 'r' varia entre 0,1 e 1, representando diferentes
             tolerâncias.
120        A mediana indica a complexidade global do padrão de
             movimento.
121        Valores mais altos de mediana sugerem maior variabilidade
             nos dados.
122
```

```
    2. **Erro Padrão SampEn**:
       O erro padrão da mediana indica a precisão dos cálculos de
          SampEn.
       Quanto menor o erro, mais confiável é a estimativa da
          mediana.

    3. **Variância SampEn**:
       A variância mede a dispersão dos valores de SampEn.
       Uma maior variância indica maior irregularidade ou
          imprevisibilidade nos dados.

    4. **Erro Normalizado SampEn**:
       O erro normalizado ajusta a precisão da mediana levando em
          conta sua magnitude.
       Um erro menor significa que a mediana é bem estimada e
          confiável.
    """)

    # Carregar os dados e calcular as estatísticas
    dados = carregar_dados(r"C:\Users\julia\Desktop\PIBIC\controle-
       postural\controle-postural\BOS_0605_CP_CV_01.txt")
    r_vals = np.linspace(0.1, 1.0, 5)  # Valores de r para análise

    if dados is not None:
        ax = dados.iloc[:, 1].values
        ay = dados.iloc[:, 2].values
        az = dados.iloc[:, 3].values

        # Inicializar dicionário para armazenar as estatísticas
        estatisticas = {'X': [], 'Y': [], 'Z': []}

        for r in r_vals:
            mse_x = multiscale_entropy(ax, scale_max=20, m=2, r=r)
            mse_y = multiscale_entropy(ay, scale_max=20, m=2, r=r)
            mse_z = multiscale_entropy(az, scale_max=20, m=2, r=r)

            estatisticas['X'].append(calcular_estatisticas_sampen(
               mse_x, mse_y))
            estatisticas['Y'].append(calcular_estatisticas_sampen(
               mse_y, mse_z))
            estatisticas['Z'].append(calcular_estatisticas_sampen(
               mse_x, mse_z))

        # Converter listas para tuplas para facilitar a
```

```
            descompactação na função de plotagem
158         estatisticas = {eixo: list(zip(*valores)) for eixo, valores
                in estatisticas.items()}
159
160         # Gerar o gráfico
161         fig = plotar_estatisticas_combinadas(r_vals, "
               BOS_0605_CP_CV_01.txt", estatisticas)
162
163         # Embutir o gráfico na interface
164         canvas = FigureCanvasTkAgg(fig, master=root)
165         canvas.get_tk_widget().grid(row=0, column=0, padx=10, pady
               =10)
166         canvas.draw()
167
168     root.mainloop()
169
170 # Iniciar a interface gráfica
171 abrir_interface()
```