vetor_RSSI_simulado

May 17, 2025

1 Simulação de Vetores RSSI com 10 APs em Área 20m x 20m

[19]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[20]:

```
# ================================
# 1. Parâmetros do ambiente
# ================================
area_size = 20
grid_step = 1
n_aps = 10
n_measurements = 10  # por ponto de treino
frequency = 2.4e9  # Hz
pt = 20  # dBm
pl0 = 40  # dB
mu = 3.5
sigma = 2
d0 = 1  # m
```

[21]:

```
# ================================
# 2. Posições fixas dos APs
# ================================
np.random.seed(42)
ap_positions = np.random.uniform(0, area_size, size=(n_aps, 2))
```

[22]:

```
# ================================
# 3. Função para gerar RSSI
# ================================
def simulate_rssi(point):
    rssi_values = []
    for ap in ap_positions:
        d = np.linalg.norm(point - ap)
        d = max(d, d0)
        path_loss = pl0 + 20 * np.log10(frequency) + 10 * mu * np.log10(d / d0)
        noise = np.random.normal(0, sigma)
        rssi = pt - path_loss + noise
        rssi = max(rssi, -110)
        rssi_values.append(rssi)
    return rssi_values
```

[43]:

```
# ================================
# 4. Posições de Treinamento (centrais em grid 10x10)
# ================================

train_grid_step = 2  # espaçamento de 2 metros → 10x10 = 100 pontos
train_x, train_y = np.meshgrid(np.arange(1, area_size, train_grid_step),
                               np.arange(1, area_size, train_grid_step))
train_positions = np.column_stack((train_x.ravel(), train_y.ravel()))


# Simular RSSI para treino
train_data = []
for pos in train_positions:
    for _ in range(n_measurements):  # 10 medições por ponto
        rssi_vector = simulate_rssi(pos)
        train_data.append(rssi_vector + [pos[0], pos[1]])
```

[44]:

```
# ================================
# 5. Posições de Teste (2 aleatórios por zona)
# ================================
test_positions = []
for i in range(area_size):
    for j in range(area_size):
        for _ in range(2):  # dois pontos por zona
            x = i + np.random.rand()
            y = j + np.random.rand()
            test_positions.append((x, y))

# Simular RSSI para teste
test_data = []
for pos in test_positions:
    rssi_vector = simulate_rssi(np.array(pos))
    test_data.append(rssi_vector + [pos[0], pos[1]])
```

[45]:

```
# ================================
# 6. DataFrames finais
# ================================
columns = [f'WAP{str(i+1).zfill(3)}' for i in range(n_aps)] + ['X', 'Y']
df_simulated = pd.DataFrame(train_data, columns=columns)
df_test = pd.DataFrame(test_data, columns=columns)

# Adicionar coluna de origem (opcional)
df_simulated["source"] = "real"
```

```
df_test["source"] = "test"
```

[46]:

```
# ================================
# 7. Unificar para uso no GAN
# ================================
df_all = pd.concat([df_simulated, df_test], ignore_index=True)
```

[47]:

```
# ================================
# 8. Verificação
# ================================
print(f"Treino: {len(df_simulated)} vetores")
print(f"Teste: {len(df_test)} vetores")
print(f"Total no dataset final (df_all): {len(df_all)} vetores")
```

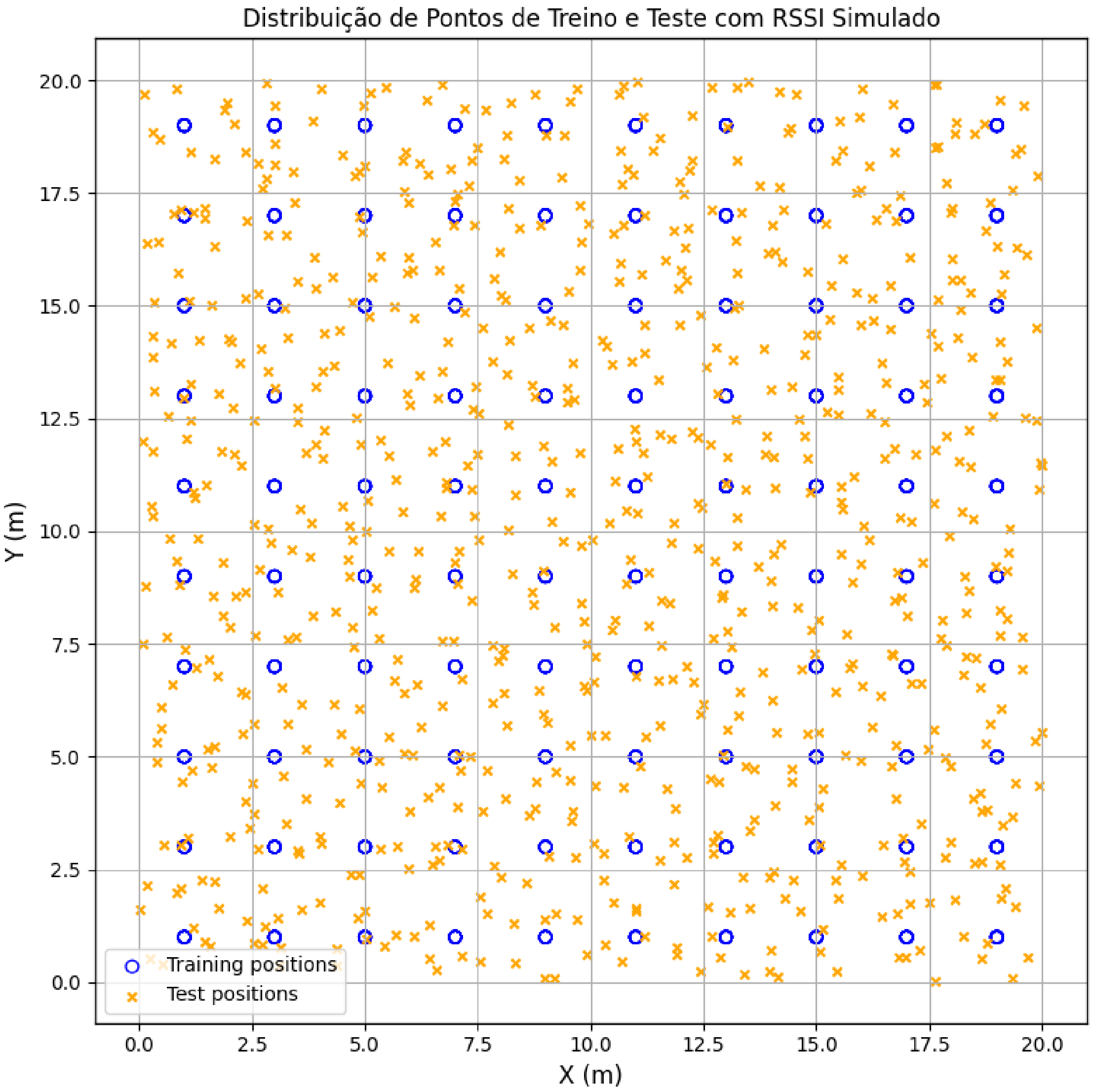
```
Treino: 1000 vetores
Teste: 800 vetores
Total no dataset final (df_all): 1800 vetores
```

[48]:

```
# ================================
# 9. Salvar CSVs (opcional)
# ================================
df_simulated.to_csv("/home/darkcover/Documentos/Gan/Data/df_simulated.csv",␣
 ↪index=False)
df_test.to_csv("/home/darkcover/Documentos/Gan/Data/df_test.csv", index=False)
df_all.to_csv("/home/darkcover/Documentos/Gan/Data/df_all.csv", index=False)
```

[51]:

```
import matplotlib.pyplot as plt

# Separar treino e teste com base na coluna "source"
df_train_vis = df_all[df_all['source'] == 'real']
df_test_vis = df_all[df_all['source'] == 'test']

# Criar figura
plt.figure(figsize=(8, 8))

# Plotar pontos de treino (azul)
plt.scatter(df_train_vis['X'], df_train_vis['Y'], marker='o', facecolors='none',
            edgecolors='blue', s=40, label='Training positions')

# Plotar pontos de teste (vermelho)
plt.scatter(df_test_vis['X'], df_test_vis['Y'],
            c='orange', marker='x', s=20, label='Test positions')

# Configurações do gráfico
plt.xlabel("X (m)")
plt.ylabel("Y (m)")
plt.title("Distribuição de Pontos de Treino e Teste com RSSI Simulado")
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.tight_layout()
plt.show()
```



[34]:
```
import matplotlib.pyplot as plt
import matplotlib as mpl

# Estilo geral
mpl.rcParams.update({
    "font.size": 10,
    "axes.labelsize": 12,
    "axes.titlesize": 12,
    "legend.fontsize": 10,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10,
    "figure.dpi": 100,
    "axes.linewidth": 1
})

# Separar treino e teste
df_train_vis = df_all[df_all['source'] == 'real']
df_test_vis = df_all[df_all['source'] == 'test']

# Criar figura
plt.figure(figsize=(6.5, 6.5))

# Test positions (vermelho escuro)
plt.scatter(df_test_vis['X'], df_test_vis['Y'],
            c='darkred', marker='x', s=40, label='Test positions')

# Training positions (azul claro com contorno)
plt.scatter(df_train_vis['X'], df_train_vis['Y'],
            facecolors='lightblue', edgecolors='black',
            marker='o', s=50, label='Training positions')

# Configurações finais
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)
plt.legend(loc='upper right', frameon=True)
plt.axis('equal')
plt.xlim(0, 20)
plt.ylim(0, 20)

# Remover título superior, adicionar título estilo legenda inferior
plt.tight_layout()
plt.subplots_adjust(bottom=0.12)
plt.figtext(0.5, 0.01, "FIGURE 3. The location of training and test positions on the simulated indoor environment.",
            wrap=True, horizontalalignment='center', fontsize=10)

plt.show()
```

```
Ignoring fixed x limits to fulfill fixed data aspect with adjustable data
limits.
Ignoring fixed y limits to fulfill fixed data aspect with adjustable data
limits.
```

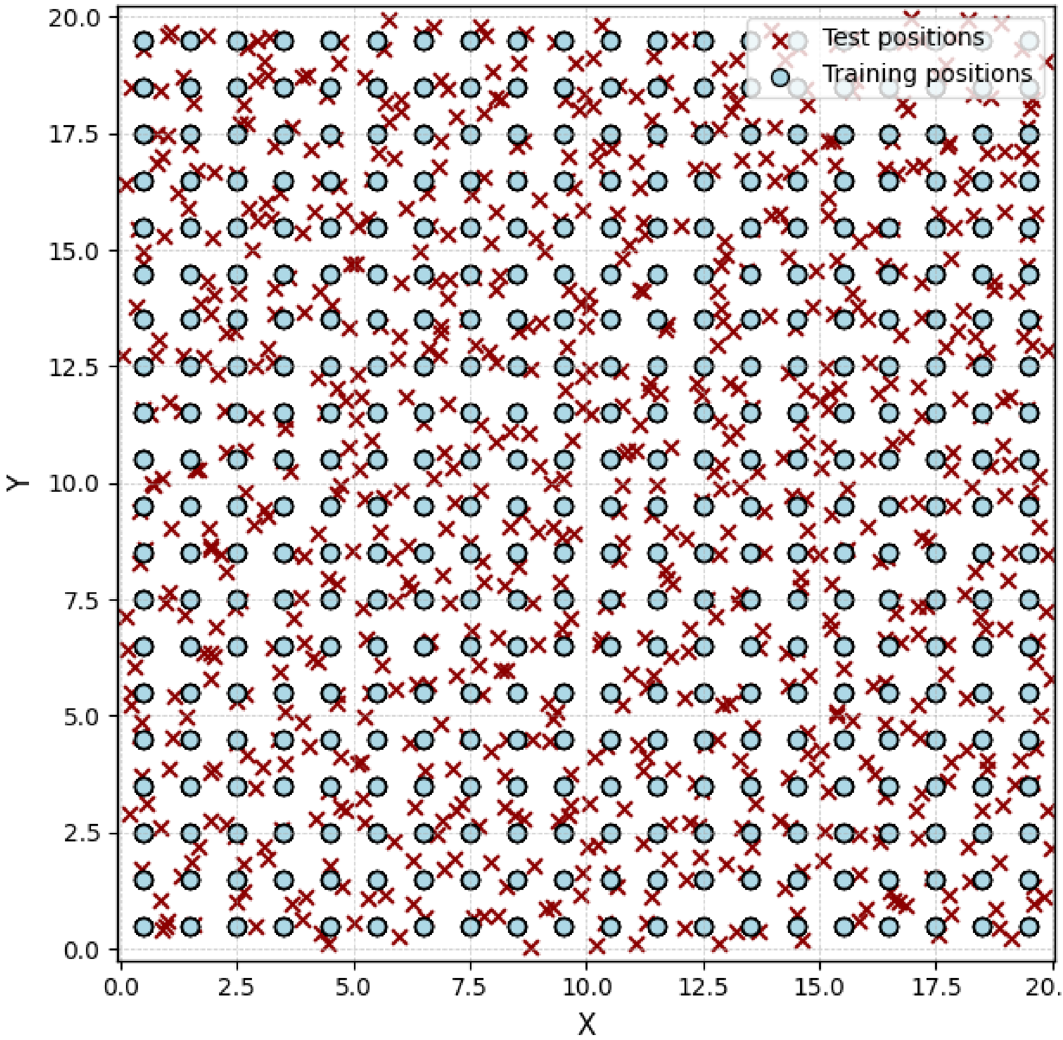


FIGURE 3. The location of training and test positions on the simulated indoor environment.