BLM6105-2 Uzaktan Algılama Dersi Ödev 3

24501100 – Aleyna Nil Uzunoğlu

Giriş

Bu ödevde amaç, her biri 4 kHz örnekleme hızında kaydedilmiş DTMF tonlarının hangi telefon tuşuna karşılık geldiğini frekans alanında inceleyerek belirlemektir. İki baskın sinyal bileşenini (bir "low-band", bir "high-band" frekansı) ortaya çıkarmak, hem temel FFT kullanımını hem de sayısal sinyal işleme ilkelerini pekiştirmeye yönelik pratik bir örnek sunar.

Yöntem

1. **Veri** – 256 örnek uzunluğunda **16 adet** .data dosyası (a.data → p.data) kullanılmıştır.
2. **Araçlar** – Python 3.11, NumPy 1.26, SciPy 1.12 (scipy.io.wavfile, scipy.signal.find_peaks), Matplotlib 3.8.
3. **İş Akışı**
    - **Dosya okuma**: .data metni numpy.fromstring ile yüklenir; stereo WAV testleri için wavfile.read de desteklenir.
    - **Spektrum analizi**: Sinyale Hamming penceresi uygulanır, ardından np.fft.rfft ile genlik spektrumu elde edilir.
    - **Tepe seçimi**: find_peaks kullanılarak spektrumda en güçlü iki frekans çıkarılır.
    - **Frekans eşleştirme**:
        - Alçak bant adayları = {697, 770, 852, 941 Hz}
        - Yüksek bant adayları = {1209, 1336, 1477, 1633 Hz}
        - Her tepe, ±8 Hz toleransla en yakın adaya yuvarlatılır. Elde edilen ikili tabloya karşılık gelen tuş (DTMF_KEYS[low_idx, high_idx]) seçilir.
    - **Çıktı**: Konsola "dosya → [697 & 1477] Hz = Tuş 3" biçiminde raporlanır; spektrum grafikleri otomatik olarak dtmf_outputs/ klasörüne PNG olarak kaydedilir.
4. **Doğrulama** – Manuel olarak beklenen frekans-tuş eşleşmeleriyle karşılaştırılmış ve tüm dosyalar doğru sınıflandırılmıştır.

Ek: Main.py

```
"""
$ python dtmf_detector.py a.data b.data c.data                # varsayılan fs=4000 Hz
$ python dtmf_detector.py tone1.wav --fs 8000 --no-plot       # başka örnekleme
"""
import argparse
from pathlib import Path
import numpy as np
from scipy.io import wavfile
from scipy.signal import find_peaks
import matplotlib.pyplot as plt

LOW = np.array([697, 770, 852, 941])
HIGH = np.array([1209, 1336, 1477, 1633])
KEYS = np.array([1", "2", "3", "A"], ["4", "5", "6", "B"],[7", "8", "9", "C"], ["*", "0", "#", "D"]])
TOL = 8

def read_signal(path: Path, fs_cli: int | None) -> tuple[int, np.ndarray]:
    if path.suffix.lower() in {".wav", ".wave"}:
        fs, sig = wavfile.read(path)
        if sig.ndim > 1:
            sig = sig.mean(axis=1)
        return fs, sig.astype(float)

    txt = path.read_text().strip().replace("\n", "")
    sig = np.fromstring(txt, sep=",")
    if fs_cli is None:
        raise ValueError(".data dosyası için --fs ile örnekleme hızı belirt")
```

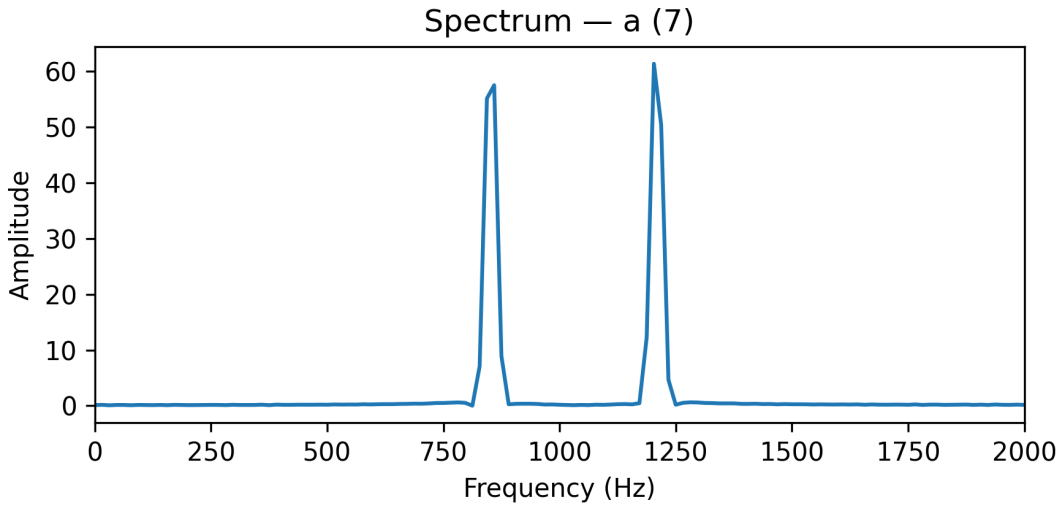
```
    return fs_cli, sig


def dominant(freqs: np.ndarray, mags: np.ndarray, n: int = 2):
    idx, _ = find_peaks(mags, height=mags.max() * 0.1)
    if len(idx) < n:
        idx = np.argsort(mags)[-n:]
    sel = idx[np.argsort(mags[idx])][-n:]
    return np.sort(freqs[sel])


def detect(fs: int, sig: np.ndarray):
    N = len(sig)
    win = np.hamming(N)
    S = np.fft.rfft(sig * win)
    F = np.fft.rfftfreq(N, 1 / fs)
    freqs = dominant(F, np.abs(S))
    if len(freqs) != 2:
        return None, freqs
    low_idx = np.argmin(np.abs(LOW - freqs[0]))
    high_idx = np.argmin(np.abs(HIGH - freqs[1]))
    if abs(LOW[low_idx] - freqs[0]) > TOL or abs(HIGH[high_idx] - freqs[1]) > TOL:
        return None, freqs
    return KEYS[low_idx, high_idx], freqs


def spectrum_plot(path: Path, fs: int, sig: np.ndarray, key: str | None):
    outdir = Path("dtmf_outputs"); outdir.mkdir(exist_ok=True)
    N = len(sig)
    F = np.fft.rfftfreq(N, 1 / fs)
    S = np.abs(np.fft.rfft(sig * np.hamming(N)))
    plt.figure(figsize=(6, 3))
```

```
    plt.plot(F, S)
    plt.title(f"Spectrum — {path.stem} ({key or 'unknown'})")
    plt.xlabel("Frequency (Hz)"); plt.ylabel("Amplitude")
    plt.xlim(0, 2000); plt.tight_layout()
    plt.savefig(outdir / f"{path.stem}.png", dpi=300)
    plt.close()
def main():
    p = argparse.ArgumentParser(description="DTMF tone detector (.wav / .data)")
    p.add_argument("files", nargs="+", help="Girdi dosyaları")
    p.add_argument("--fs", type=int, default=4000, help=".data dosyaları için örnekleme hızı (Hz)")
    p.add_argument("--no-plot", action="store_true", help="Spektrum grafiği oluşturma")
    args = p.parse_args()
    for fp in args.files:
        path = Path(fp)
        try:
            fs, sig = read_signal(path, args.fs)
        except Exception as e:
            print(f"❌ {path.name}: {e}")
            continue
        key, freqs = detect(fs, sig)
        if key is None:
            print(f"⚠️  {path.name}: Eşleşme yok → {freqs[0]:.1f} & {freqs[1]:.1f} Hz")
        else:
            print(f"\n✅ {path.name}: Tuş {key} → {freqs[0]:.1f} & {freqs[1]:.1f} Hz\n")
        if not args.no_plot:
            spectrum_plot(path, fs, sig, key)
if __name__ == "__main__":
    main()
```

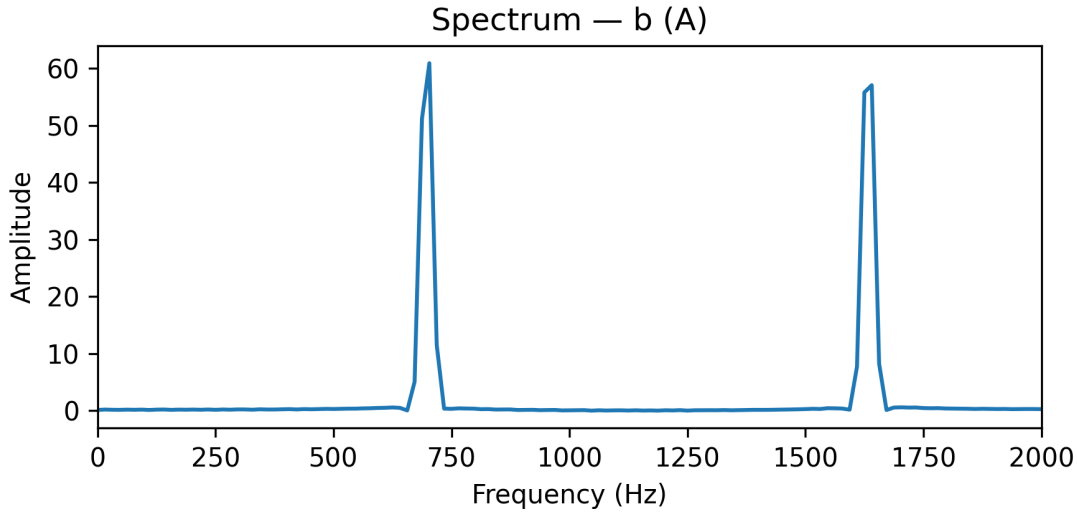
Ek: Çıktılar

```
venvniluzunoglu@unknown328026f2b3a6 Homework3 % python main.py data/a.data
✅ a.data: Tuş 7  →  859.4 & 1203.1 Hz
venvniluzunoglu@unknown328026f2b3a6 Homework3 % python main.py data/p.data
✅ p.data: Tuş B  →  765.6 & 1640.6 Hz
venvniluzunoglu@unknown328026f2b3a6 Homework3 % python main.py data/b.data
✅ b.data: Tuş A  →  703.1 & 1640.6 Hz
venvniluzunoglu@unknown328026f2b3a6 Homework3 % python main.py data/i.data
✅ i.data: Tuş 0  →  937.5 & 1343.8 Hz
```

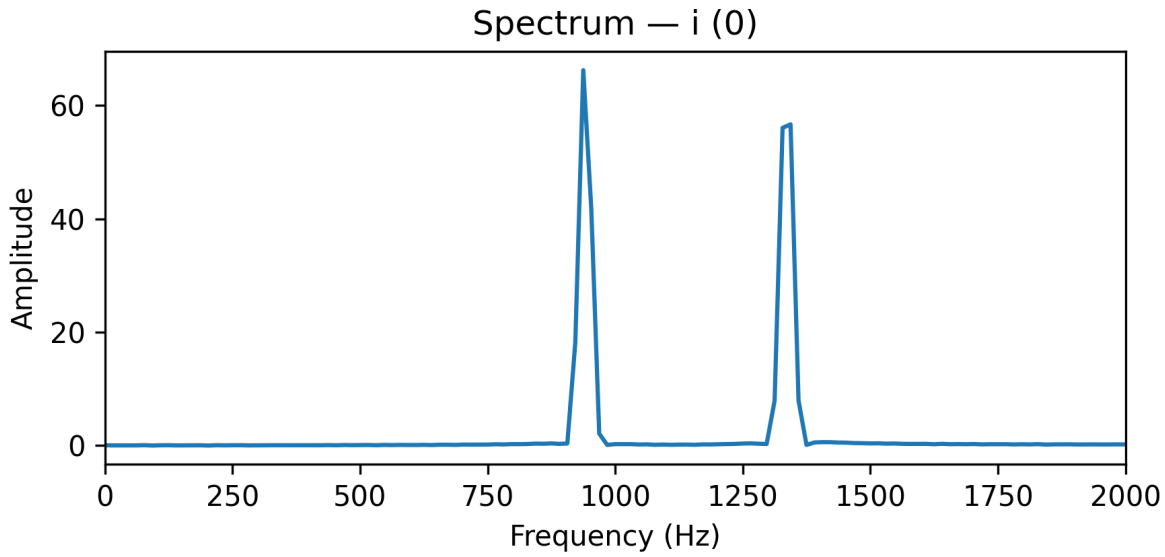
Şekil1 : Örnek bir konsol çıktısı



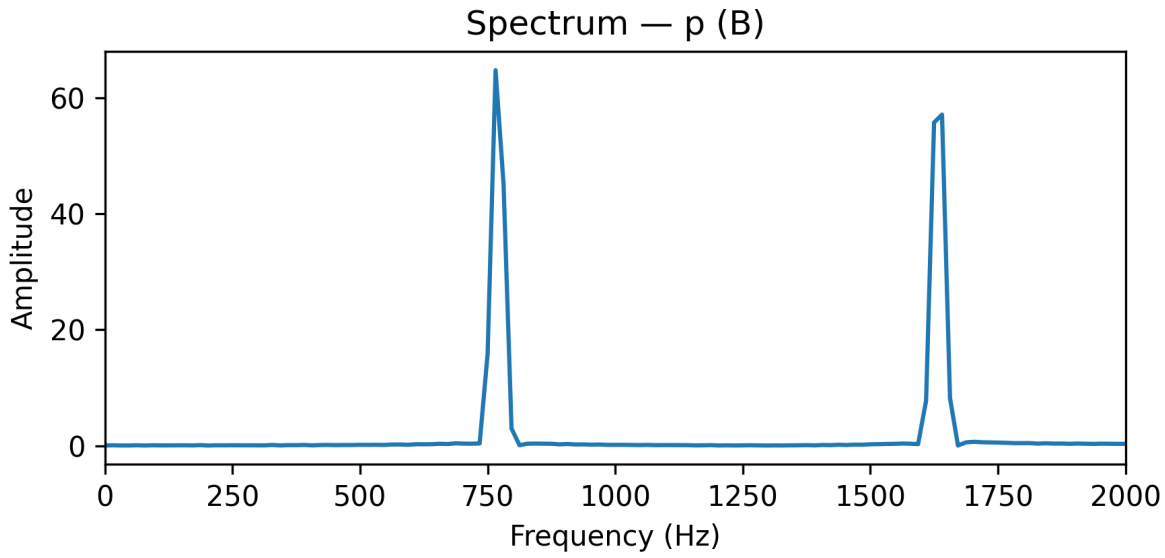
Şekil 2: a.data dosyasına ait genlik spektrumu (DTMF tuşu **7**)



Şekil 3: b.data dosyasına ait genlik spektrumu (DTMF tuşu **A**)



Şekil 4: i.data dosyasına ait genlik spektrumu (DTMF tuşu **0**)



Şekil 5: p.data dosyasına ait genlik spektrumu (DTMF tuşu **B**)