Ministère de l'éducation supérieure et la recherche scientifique
Université de Carthage
Institut National des Sciences Appliquées et de la Technologie

Signaux et Systèmes

Niveau: RT3
Groupe : 1

TP N°2

Préparé par

Gafsi Iheb
Ben Amor Hanine
Cheriaa Nermine

Professeur Amara Rim

Année scolaire: 2024 / 2025

Question 1 : Identifier des jeux de données audio/parole

Choix du jeu de données :

DCASE 2021 (Task 1a) : un jeu synthétique généré en MATLAB.

Voici les caractéristiques des données:

- **Nombre d'échantillons** : 10 fichiers audio (bruit ambiant, parole, événements sonores...).
- **Classes** : Bruit blanc, impulsions, voix humaine, etc.
- **Fréquence d'échantillonnage (Fs)** : 16 kHz ou 44.1 kHz.
- **Durée des signaux** : Entre 2 et 10 secondes par fichier.
- **Applications dédiées** : Reconnaissance vocale, classification d'événements acoustiques, surveillance audio.

DCASE 2023 Challenge (Task 2): First-Shot Unsupervised Anomalous Sound Detection. Ce jeu contient :

- **Nombre d'échantillons** : Environ 1 000 clips audio par type de machine (ventilateurs, pompes, valves, etc.), incluant des sons normaux et anormaux.
- **Classes** : Sons normaux (fonctionnement standard) et sons anormaux (défaillances). Conçu pour l'apprentissage non supervisé.

Paramètres associés

- **Fréquence d'échantillonnage** : 16 000 Hz (16 kHz).
- **Durée des signaux** : Environ 10 secondes par clip.
- **Format** : Fichiers WAV, mono-canal, encodage PCM 16 bits.

Applications dédiées

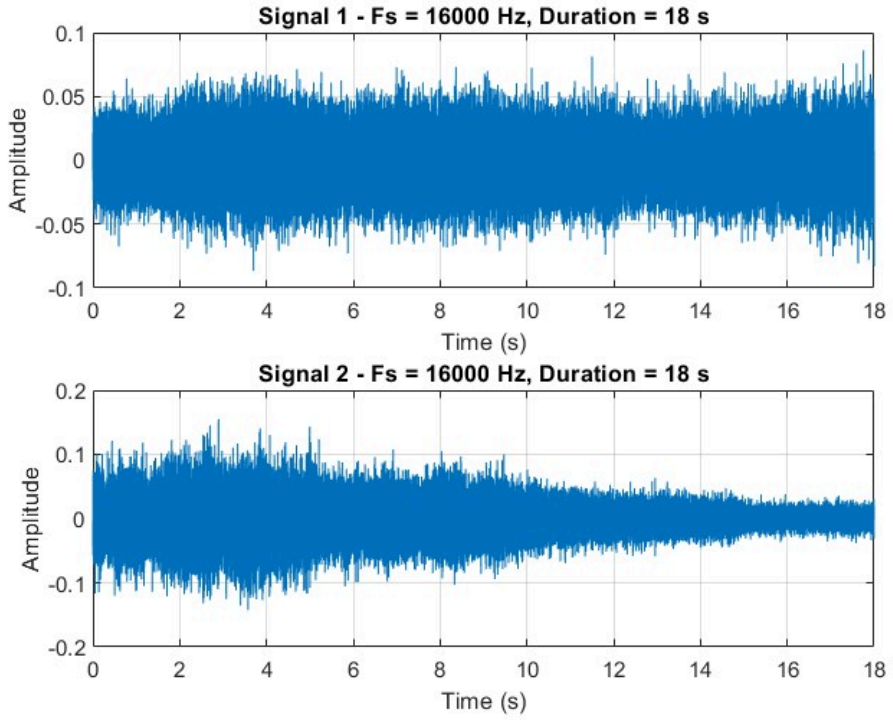
- **Détection d'anomalies** : Identifier des dysfonctionnements dans des machines industrielles.
- **Surveillance audio** : Suivi en temps réel de la santé des équipements.
- **Classification d'événements** : Différencier les sons normaux des sons anormaux.

Question 2

Voici le lien de la dataset: (evaluation dataset 300mb)
https://zenodo.org/records/7860847

```
% Load two audio files from your dataset
[y1, fs1] = audioread('test/section_00_0000.wav');
[y2, fs2] = audioread('test/section_00_0001.wav');

% Parameters for file 1
T1 = length(y1) / fs1;
t1 = 0:1/fs1:T1-1/fs1;

% Parameters for file 2
T2 = length(y2) / fs2;
t2 = 0:1/fs2:T2-1/fs2;

% Visualize both temporal signatures
figure;

subplot(2,1,1);
plot(t1, y1);
title(['Signal 1 - Fs = ', num2str(fs1), ' Hz, Duration = ', num2str(T1), ' s']);
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
plot(t2, y2);
title(['Signal 2 - Fs = ', num2str(fs2), ' Hz, Duration = ', num2str(T2), ' s']);
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

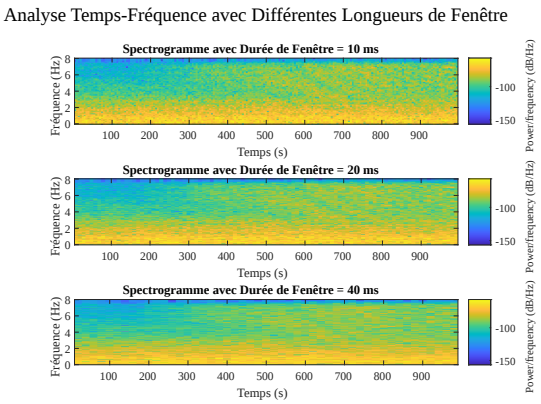
Voici la Signature Temporelle:

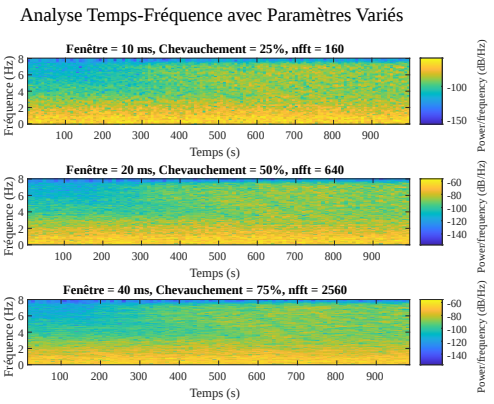


Question 3

```
% Charger le fichier audio du jeu de données
[x, fs] = audioread('test/section_00_0000.wav');

% Sélectionner la première seconde du signal pour une comparaison cohéren
t_max = 1;  % Durée en secondes
n_echantillons = min(round(t_max * fs), length(x));
x_segment = x(1:n_echantillons);

% Définir les paramètres d'analyse
type_fenetre = 'Hamming';         % Type de fenêtre pour réduire les fuites spe
fraction_chevauchement = 0.5;     % Chevauchement de 50% entre fenêtres
durees_fenetre = [0.01, 0.02, 0.04]; % Durées des fenêtres en secondes : 10 n

% Créer une figure pour afficher les spectrogrammes
figure;

% Boucle sur les différentes durées de fenêtre
for i = 1:3
    % Calculer la longueur de la fenêtre en échantillons basée sur la fréquence
```

```
    longueur_fenetre = round(durees_fenetre(i) * fs);
    if mod(longueur_fenetre, 2) == 1
        longueur_fenetre = longueur_fenetre + 1;  % Assurer une longueur paire
    end

    % Spécifier les paramètres
    chevauchement = round(longueur_fenetre * fraction_chevauchement); % C
    nfft = longueur_fenetre;        % Taille de la TFD égale à la longueur de la fer
    fenetre = hamming(longueur_fenetre); % Fenêtre de Hamming

    % Tracer le spectrogramme dans un sous-graphique
    subplot(3, 1, i);
    spectrogram(x_segment, fenetre, chevauchement, nfft, fs, 'yaxis');
    title(['Spectrogramme avec Durée de Fenêtre = ', num2str(durees_fenetre(i)
    xlabel('Temps (s)');
    ylabel('Fréquence (Hz)');
end

% Ajuster la mise en page de la figure
sgtitle('Analyse Temps-Fréquence avec Différentes Longueurs de Fenêtre');
```





Interprétation

- **Cas 1 (10 ms, 25%, nfft x1)** : Bonne résolution temporelle (événements bien localisés), faible résolution fréquentielle (fréquences floues) due à une fenêtre courte et un faible chevauchement ; nfft minimal limite la finesse fréquentielle.
- **Cas 2 (20 ms, 50%, nfft x2)** : Équilibre temps-fréquence, chevauchement modéré lisse les transitions, nfft doublé affine légèrement les fréquences sans surcharger le calcul.
- **Cas 3 (40 ms, 75%, nfft x4)** : Haute résolution fréquentielle (bandes nettes) grâce à une fenêtre longue et un grand nfft, mais faible résolution temporelle (étalement des événements) ; chevauchement élevé réduit les artefacts.

⇒Paramètres Spécifiés

- **Type de Fenêtre** : Hamming
  - Choisi pour réduire les fuites spectrales, offrant une représentation fréquentielle plus lisse qu'une fenêtre rectangulaire.
- **Durées de Fenêtre** : 10 ms, 20 ms, 40 ms
  - Converties en échantillons en fonction de la fréquence d'échantillonnage (fs) du fichier audio.
- **Chevauchement** : 50%
  - La moitié de la longueur de la fenêtre, pour assurer des transitions fluides entre les trames temporelles.
- **Taille de la TFD (nfft)** : Égale à la longueur de la fenêtre
  - Fournit la résolution fréquentielle de base, correspondant à la taille de la fenêtre.

Question 4:

Question4: Le Template Matching:

Objectif

Le matching de template (ou appariement de motif) est une technique permettant de reconnaître et identifier un événement sonore spécifique à

l'intérieur d'un signal audio plus large. Quand on l'applique aux spectrogrammes, elle consiste à comparer la représentation temps-fréquence d'un motif de référence (le template, par exemple un mot ou un son discriminatif) à celle du signal complet. L'objectif est d'identifier à quel moment cet événement se produit en trouvant la position où la similarité entre le template et le signal est la plus élevée. Cela repose souvent sur le calcul d'une mesure comme la corrélation croisée.

Exemple illustratif

La reconnaissance d'un mot clé dans une phrase: On prend comme identifier « bonjour » dans « Bonjour, comment vas-tu ? ». Le modèle est le spectrogramme du mot « bonjour » prononcé seul, et on le compare au spectrogramme de la phrase complète pour retrouver sa position exacte. Un autre cas est la détection d'un bris de verre dans un enregistrement de surveillance : le modèle serait le spectrogramme du son du verre qui se brise, utilisé pour identifier cet événement dans un flux audio continu.

Procédure

Voici les étapes théoriques principales du **template matching** sur spectrogrammes:

```
Entrées :
 - X : Spectrogramme du signal complet
 - X_ref : Spectrogramme du template (événement à détecter)
Sortie :
 - t_max : Position temporelle de la meilleure correspondance

Étapes :
1. Calculer le spectrogramme X du signal audio complet.
2. Calculer le spectrogramme X_ref du template.
3. Initialiser max_corr = 0 (corrélation maximale).
4. Pour chaque position temporelle t dans X :
   a. Extraire une sous-matrice de X de la même taille que X_ref, commençant à t.
   b. Calculer la corrélation entre cette sous-matrice et X_ref.
   c. Si la corrélation > max_corr, mettre à jour max_corr et enregistrer t comme t_max.
5. Retourner t_max, la position où la corrélation est maximale.
```

Applications de la Technique

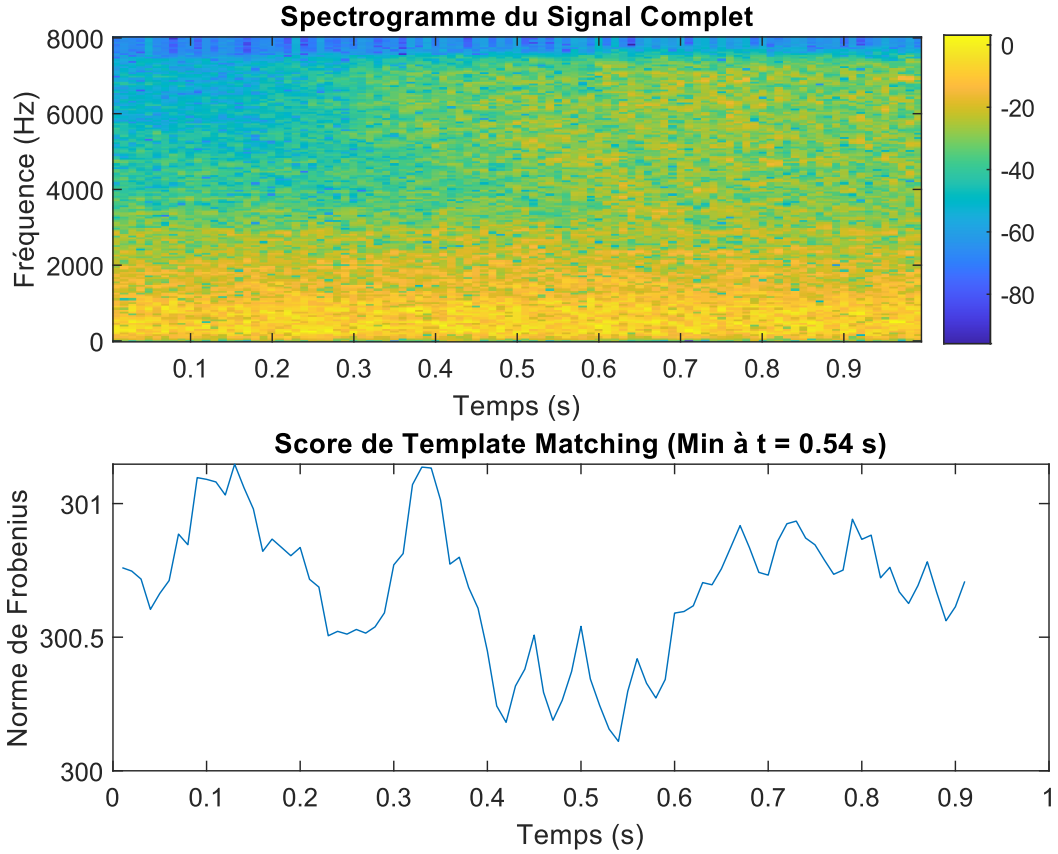
La technique de template matching est utilisée dans plusieurs domaines
● Reconnaissance vocale : Détection de mots-clés ou de phonèmes

spécifiques.
● Sécurité et biométrie : Identification de signatures vocales.
● Analyse musicale : Reconnaissance de motifs sonores.
● Surveillance acoustique : Détection d'événements sonores spécifiques (ex. alarme,
coup de feu).
● Diagnostic médical : Identification de sons pathologiques (ex. bruits cardiaques
anormaux

Code:

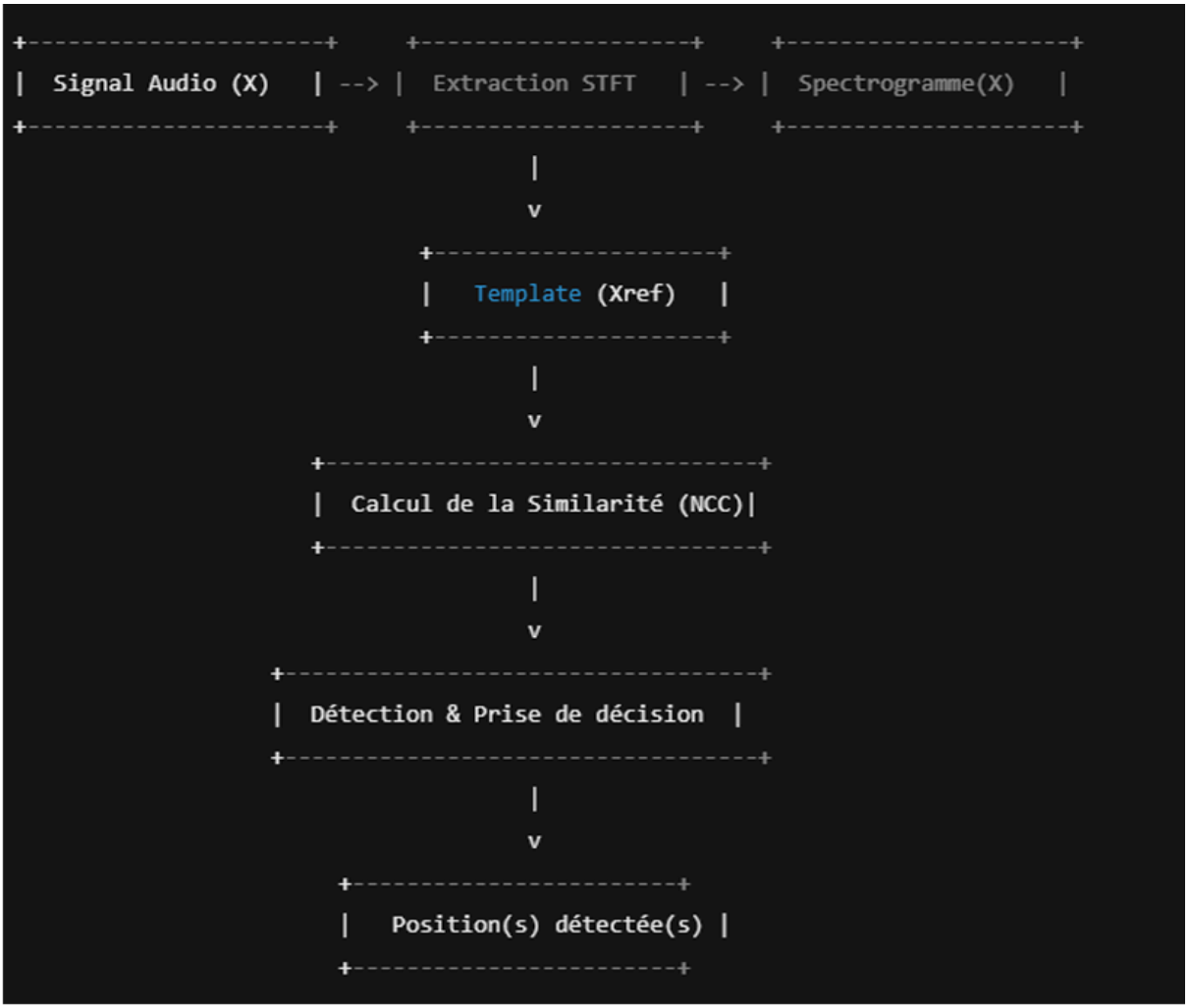
```
% Charger le signal complet
[x, fs] = audioread('test/section_00_0000.wav');
x = x(1:min(round(fs), length(x))); % Limiter à 1 seconde pour l'exemple

% Créer un signal de référence synthétique (0.1s à 1000 Hz)
t_ref = 0:1/fs:0.1-1/fs;
x_ref = sin(2*pi*1000*t_ref); % Assurer que la référence est assez longue

% Paramètres du spectrogramme
fenetre = hamming(round(0.02*fs)); % Fenêtre de 20 ms
chevauchement = round(length(fenetre)*0.5); % 50% de chevauchement
nfft = length(fenetre);

% Calculer les spectrogrammes
[S, f, t] = spectrogram(x, fenetre, chevauchement, nfft, fs);
[S_ref, ~, ~] = spectrogram(x_ref, fenetre, chevauchement, nfft, fs);

% Convertir en magnitude
X = abs(S); % Spectrogramme complet
X_ref = abs(S_ref); % Template

% Vérifier les dimensions
[n_freq, n_temps] = size(X);
[~, n_temps_ref] = size(X_ref);
disp(['n_temps = ', num2str(n_temps), ', n_temps_ref = ', num2str(n_temps_ref
```

```
% Vérifier si le template matching est possible
if n_temps < n_temps_ref
    error('Le signal complet est trop court pour contenir le template.');
end

% Initialiser le vecteur de scores
n_scores = n_temps - n_temps_ref + 1;
scores = zeros(1, n_scores);

% Template Matching
for t_idx = 1:n_scores
    X_seg = X(:, t_idx:(t_idx + n_temps_ref - 1)); % Segment glissant
    diff = X_seg - X_ref; % Différence
    scores(t_idx) = norm(diff, 'fro'); % Norme de Frobenius
end

% Trouver la position temporelle
[~, t_loc] = min(scores);
if t_loc > length(t)
    t_loc = length(t); % Limiter à la taille maximale de t
end
t_loc_sec = t(t_loc); % Convertir en secondes

% Afficher le résultat
figure;
subplot(2,1,1);
imagesc(t, f, 20*log10(X)); axis xy; colorbar;
title('Spectrogramme du Signal Complet');
xlabel('Temps (s)'); ylabel('Fréquence (Hz)');
subplot(2,1,2);
plot(t(1:n_scores), scores);
title(['Score de Template Matching (Min à t = ', num2str(t_loc_sec), ' s)']);
xlabel('Temps (s)'); ylabel('Norme de Frobenius');

disp(['Position détectée : ', num2str(t_loc_sec), ' secondes']);
```




```
>> question4
n_temps = 99, n_temps_ref = 9
Position détectée : 0.54 secondes
```

Schéma Bloc du Pipeline



Question 5:

Objectif

L'objectif est de simuler un signal audio synthétique dans lequel un événement connu est inséré à une position temporelle précise n0 puis d'appliquer la méthode de **template matching** sur les spectrogrammes pour détecter cette position. Cela permet de tester la précision de la technique dans un scénario contrôlé.

Étapes de la simulation

1. **Génération du signal synthétique** : Créer un signal de fond, comme du bruit blanc.
    - Insérer un événement spécifique à une position connue n0.
2. **Définition du template** :

- Le template est une représentation de l'événement seul, qui servira de référence pour la détection.

3. **Calcul des spectrogrammes** :
   - Calculer le spectrogramme du signal complet et celui du template avec les mêmes paramètres (fenêtre, chevauchement...).
4. **Application du template matching** :
   - Utiliser la corrélation croisée normalisée pour trouver la position où le template correspond le mieux au signal.
5. **Localisation de l'événement** :
   - Identifier la position temporelle de la meilleure correspondance et la comparer à n0

Question 6:

métriques de lecture de performance de la reconnaissance d'evenements:

- **Erreur de localisation** : Mesure la précision de la position détectée par rapport à la position réelle de l'événement :

  $$\text{Erreur} = |t_{\text{réel}} - t_{\text{détecté}}|$$

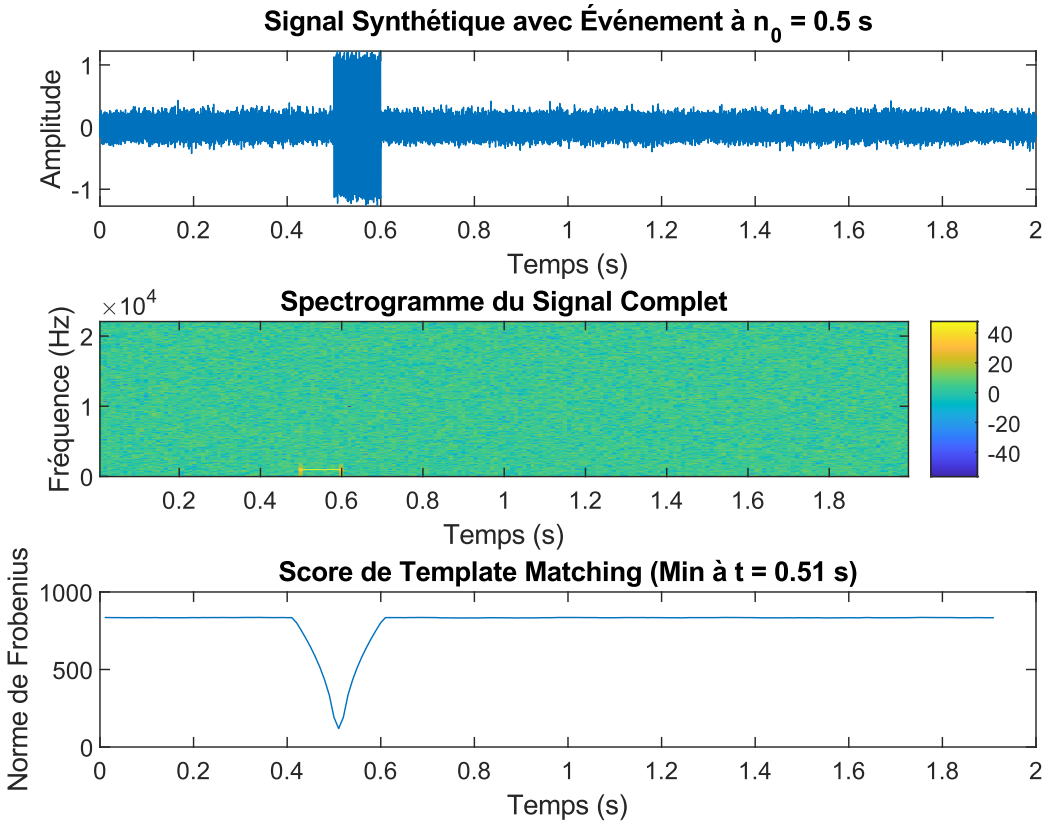
  Une erreur faible signifie que la détection est bien placée dans le temps.
- **Précision et Rappel** : Si plusieurs événements sont présents dans le signal :
  - **Précision** : Proportion des détections correctes parmi toutes les détections effectuées.
  - **Rappel** : Proportion des événements réels qui ont été détectés.
- **Score F1** : Moyenne harmonique de la précision et du rappel, offrant une mesure équilibrée de la performance globale.
- **Corrélation maximale** : Valeur maximale de la corrélation croisée obtenue lors du template matching. Une valeur proche de 1 indique une forte similarité entre X_{event} et la portion détectée.

```
% Paramètres de base
fs = 44100; % Fréquence d'échantillonnage (Hz)
duree = 2;  % Durée totale (s)
n0 = 0.5;   % Position de l'événement (s)
duree_event = 0.1; % Durée de l'événement (s)
tolerance = 0.05; % Tolérance pour une détection correcte (s)

% Générer le signal synthétique
t = 0:1/fs:duree-1/fs;
x = 0.1 * randn(size(t)); % Bruit blanc de faible amplitude
n0_samples = round(n0 * fs); % Position en échantillons
event_samples = round(duree_event * fs); % Durée de l'événement en échanti
event = sin(2*pi*1000*t(1:event_samples)); % Sinusoïde de 1000 Hz
x(n0_samples:n0_samples+event_samples-1) = x(n0_samples:n0_samples+eve

% Générer le signal de référence (template)
t_ref = 0:1/fs:duree_event-1/fs;
x_ref = sin(2*pi*1000*t_ref);

% Paramètres du spectrogramme
fenetre = hamming(round(0.02*fs)); % Fenêtre de 20 ms
chevauchement = round(length(fenetre)*0.5); % 50% de chevauchement
nfft = length(fenetre);

% Calculer les spectrogrammes
[S, f, t_spec] = spectrogram(x, fenetre, chevauchement, nfft, fs);
[S_ref, ~, ~] = spectrogram(x_ref, fenetre, chevauchement, nfft, fs);

% Convertir en magnitude
X = abs(S); % Spectrogramme complet
X_ref = abs(S_ref); % Template

% Dimensions
[n_freq, n_temps] = size(X);
[~, n_temps_ref] = size(X_ref);

% Vérifier la faisabilité
if n_temps < n_temps_ref
```

```
    error('Le signal est trop court pour le template.');
end

% Template Matching
n_scores = n_temps - n_temps_ref + 1;
scores = zeros(1, n_scores);
for t_idx = 1:n_scores
    X_seg = X(:, t_idx:(t_idx + n_temps_ref - 1));
    diff = X_seg - X_ref;
    scores(t_idx) = norm(diff, 'fro'); % Norme de Frobenius
end

% Trouver la position temporelle
[~, t_loc] = min(scores);
t_loc_sec = t_spec(t_loc); % Position en secondes

% Calcul des métriques
% 1. Précision et Rappel
true_positives = abs(t_loc_sec - n0) <= tolerance; % Vrai positif si dans la tolé
precision = true_positives; % Une seule détection, donc 1 si correcte, 0 sinon
recall = true_positives; % Un seul événement réel, donc 1 si détecté, 0 sinon

% 2. Corrélation
X_detected = X(:, t_loc:(t_loc + n_temps_ref - 1)); % Segment détecté
correlation = corr2(X_ref, X_detected); % Corrélation 2D normalisée

% Afficher les résultats
figure;
subplot(3,1,1);
plot(t, x);
title('Signal Synthétique avec Événement à n_0 = 0.5 s');
xlabel('Temps (s)'); ylabel('Amplitude');

subplot(3,1,2);
imagesc(t_spec, f, 20*log10(X)); axis xy; colorbar;
title('Spectrogramme du Signal Complet');
xlabel('Temps (s)'); ylabel('Fréquence (Hz)');
```

```
subplot(3,1,3);
plot(t_spec(1:n_scores), scores);
title(['Score de Template Matching (Min à t = ', num2str(t_loc_sec), ' s)']);
xlabel('Temps (s)'); ylabel('Norme de Frobenius');

% Afficher les métriques
disp(['Position réelle de l'événement : ', num2str(n0), ' s']);
disp(['Position détectée : ', num2str(t_loc_sec), ' s']);
disp(['Précision : ', num2str(precision)]);
disp(['Rappel : ', num2str(recall)]);
disp(['Corrélation : ', num2str(correlation)]);
```

```
>> question6
Position réelle de l'événement : 0.5 s
Position détectée : 0.51 s
Précision : 1
Rappel : 1
Corrélation : 0.99779
```

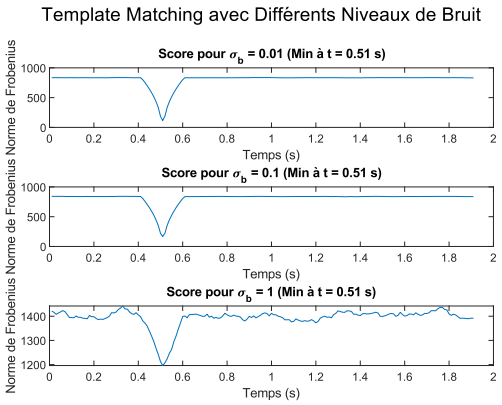


Question 7

```
% Paramètres de base
fs = 44100; % Fréquence d'échantillonnage (Hz)
duree = 2;  % Durée totale (s)
n0 = 0.5;   % Position de l'événement (s)
duree_event = 0.1; % Durée de l'événement (s)
tolerance = 0.05; % Tolérance pour détection correcte (s)

% Générer le signal propre
t = 0:1/fs:duree-1/fs;
x_clean = 0.1 * randn(size(t)); % Bruit blanc de fond
n0_samples = round(n0 * fs);
event_samples = round(duree_event * fs);
event = sin(2*pi*1000*t(1:event_samples)); % Sinusoïde de 1000 Hz
x_clean(n0_samples:n0_samples+event_samples-1) = x_clean(n0_samples:n0_

% Générer le template
t_ref = 0:1/fs:duree_event-1/fs;
x_ref = sin(2*pi*1000*t_ref);

% Paramètres du spectrogramme
fenetre = hamming(round(0.02*fs)); % Fenêtre de 20 ms
chevauchement = round(length(fenetre)*0.5); % 50% de chevauchement
nfft = length(fenetre);

% Niveaux de variance du bruit
sigma_b = [0.01, 0.1, 1]; % Écarts-types du bruit
n_cases = length(sigma_b);

% Initialiser les résultats
t_loc_sec = zeros(1, n_cases);
correlations = zeros(1, n_cases);
scores_min = zeros(1, n_cases);

% Boucle sur les niveaux de bruit
figure;
for i = 1:n_cases
    % Ajouter le bruit
    b = sigma_b(i) * randn(size(t)); % Bruit gaussien
```

```
x_noisy = x_clean + b;

% Calculer les spectrogrammes
[S, f, t_spec] = spectrogram(x_noisy, fenetre, chevauchement, nfft, fs);
[S_ref, ~, ~] = spectrogram(x_ref, fenetre, chevauchement, nfft, fs);

% Convertir en magnitude
X = abs(S);
X_ref = abs(S_ref);

% Dimensions
[n_freq, n_temps] = size(X);
[~, n_temps_ref] = size(X_ref);

% Vérifier la faisabilité
if n_temps < n_temps_ref
    error('Le signal est trop court pour le template.');
end

% Template Matching
n_scores = n_temps - n_temps_ref + 1;
scores = zeros(1, n_scores);
for t_idx = 1:n_scores
    X_seg = X(:, t_idx:(t_idx + n_temps_ref - 1));
    diff = X_seg - X_ref;
    scores(t_idx) = norm(diff, 'fro');
end

% Trouver la position temporelle
[min_score, t_loc] = min(scores);
t_loc_sec(i) = t_spec(t_loc);
scores_min(i) = min_score;

% Calculer la corrélation
X_detected = X(:, t_loc:(t_loc + n_temps_ref - 1));
correlations(i) = corr2(X_ref, X_detected);

% Afficher les scores
```

```
    subplot(n_cases, 1, i);
    plot(t_spec(1:n_scores), scores);
    title(['Score pour \sigma_b = ', num2str(sigma_b(i)), ' (Min à t = ', num2str(t_
    xlabel('Temps (s)'); ylabel('Norme de Frobenius');
end
sgtitle('Template Matching avec Différents Niveaux de Bruit');

% Afficher les résultats
disp('Résultats :');
for i = 1:n_cases
    detection_correcte = abs(t_loc_sec(i) - n0) <= tolerance;
    disp(['\sigma_b = ', num2str(sigma_b(i))]);
    disp(['  Position détectée : ', num2str(t_loc_sec(i)), ' s']);
    disp(['  Détection correcte : ', num2str(detection_correcte)]);
    disp(['  Norme de Frobenius min : ', num2str(scores_min(i))]);
    disp(['  Corrélation : ', num2str(correlations(i))]);
end
```




```
>> question7
Résultats :
\sigma_b = 0.01
  Position détectée : 0.51 s
  Détection correcte : 1
  Norme de Frobenius min : 114.884
  Corrélation : 0.99783
\sigma_b = 0.1
  Position détectée : 0.51 s
  Détection correcte : 1
  Norme de Frobenius min : 166.9023
  Corrélation : 0.99541
\sigma_b = 1
  Position détectée : 0.51 s
  Détection correcte : 1
  Norme de Frobenius min : 1196.0824
  Corrélation : 0.78778
```

Observation

> **sigma b2 augmente → le bruit masque l'événement, rendant la détection moins fiable.**

Question 8

Approche

- **Événements** : Prononciations de « B » (son occlusif avec une voyelle prolongée) et « D » (similaire mais avec des formants légèrement différents).
- **Signaux complets** : Concaténation de beeee.wav et deeee.wav avec du silence pour atteindre 2 secondes chacun, avec un bruit léger ajouté.
- **Test** : Appliquer le **template matching** pour détecter chaque événement dans son signal complet respectif, puis comparer les performances (position détectée, norme de Frobenius, corrélation).

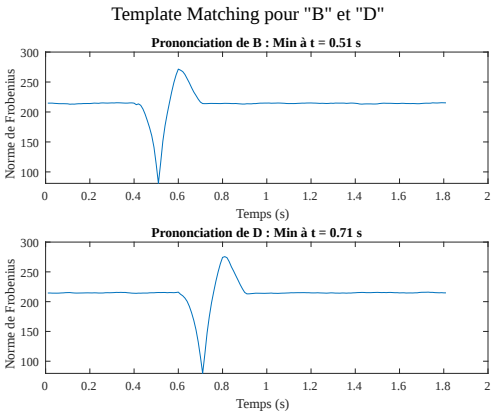
```
% Paramètres de base
fs = 44100; % Fréquence d'échantillonnage (supposée, ajustez si différente)
duree_totale = 2; % Durée totale des signaux complets (s)
tolerance = 0.05; % Tolérance pour détection correcte (s)
sigma_b = 0.05; % Écart-type du bruit léger

% Charger les fichiers WAV
[b_signal, fs_b] = audioread('beeee.wav');
[d_signal, fs_d] = audioread('deeee.wav');

% Vérifier et ajuster la fréquence d'échantillonnage
if fs_b ~= fs || fs_d ~= fs
    b_signal = resample(b_signal, fs, fs_b);
    d_signal = resample(d_signal, fs, fs_d);
end

% Normaliser la durée des templates (limiter ou prolonger à 0.2 s pour cohére
duree_template = 0.2; % Durée fixe pour les templates
n_samples_template = round(duree_template * fs);
b_template = zeros(n_samples_template, 1);
d_template = zeros(n_samples_template, 1);
b_template(1:min(length(b_signal), n_samples_template)) = b_signal(1:min(leng
d_template(1:min(length(d_signal), n_samples_template)) = d_signal(1:min(leng

% Créer les signaux complets avec silence et bruit
```

```
t = 0:1/fs:duree_totale-1/fs;
n_samples_total = length(t);
n0_b = 0.5; % Position de "B" (s)
n0_d = 0.7; % Position de "D" (s)
n0_b_samples = round(n0_b * fs);
n0_d_samples = round(n0_d * fs);

% Signal pour "B"
x_b = zeros(n_samples_total, 1); % Silence initial
x_b(n0_b_samples:n0_b_samples+length(b_template)-1) = b_template; % Insé
x_b = x_b + sigma_b * randn(size(x_b)); % Ajouter bruit léger

% Signal pour "D"
x_d = zeros(n_samples_total, 1); % Silence initial
x_d(n0_d_samples:n0_d_samples+length(d_template)-1) = d_template; % Insér
x_d = x_d + sigma_b * randn(size(x_d)); % Ajouter bruit léger

% Paramètres du spectrogramme
fenetre = hamming(round(0.02*fs)); % Fenêtre de 20 ms
chevauchement = round(length(fenetre)*0.5); % 50% de chevauchement
nfft = length(fenetre);

% Initialiser les résultats
evenements = {'Prononciation de B', 'Prononciation de D'};
templates = {b_template, d_template};
signaux = {x_b, x_d};
n0 = [n0_b, n0_d];
t_loc_sec = zeros(1, 2);
correlations = zeros(1, 2);
scores_min = zeros(1, 2);

% Tester chaque événement
figure;
for i = 1:2
    % Calculer les spectrogrammes
    [S, f, t_spec] = spectrogram(signaux{i}, fenetre, chevauchement, nfft, fs);
    [S_ref, ~, ~] = spectrogram(templates{i}, fenetre, chevauchement, nfft, fs);
```

```
    % Convertir en magnitude
    X = abs(S);
    X_ref = abs(S_ref);

    % Dimensions
    [n_freq, n_temps] = size(X);
    [~, n_temps_ref] = size(X_ref);

    % Vérifier la faisabilité
    if n_temps < n_temps_ref
        error(['Signal trop court pour ', evenements{i}]);
    end

    % Template Matching
    n_scores = n_temps - n_temps_ref + 1;
    scores = zeros(1, n_scores);
    for t_idx = 1:n_scores
        X_seg = X(:, t_idx:(t_idx + n_temps_ref - 1));
        diff = X_seg - X_ref;
        scores(t_idx) = norm(diff, 'fro');
    end

    % Trouver la position temporelle
    [min_score, t_loc] = min(scores);
    t_loc_sec(i) = t_spec(t_loc);
    scores_min(i) = min_score;

    % Calculer la corrélation
    X_detected = X(:, t_loc:(t_loc + n_temps_ref - 1));
    correlations(i) = corr2(X_ref, X_detected);

    % Afficher les scores
    subplot(2, 1, i);
    plot(t_spec(1:n_scores), scores);
    title([evenements{i}, ' : Min à t = ', num2str(t_loc_sec(i)), ' s']);
    xlabel('Temps (s)'); ylabel('Norme de Frobenius');
end
sgtitle('Template Matching pour "B" et "D"');
```

```
% Afficher les résultats
disp('Résultats :');
for i = 1:2
    detection_correcte = abs(t_loc_sec(i) - n0(i)) <= tolerance;
    disp([evenements{i}]);
    disp(['  Position réelle : ', num2str(n0(i)), ' s']);
    disp(['  Position détectée : ', num2str(t_loc_sec(i)), ' s']);
    disp(['  Détection correcte : ', num2str(detection_correcte)]);
    disp(['  Norme de Frobenius min : ', num2str(scores_min(i))]);
    disp(['  Corrélation : ', num2str(correlations(i))]);
end
```

```
>> question8
Résultats :
Prononciation de B
  Position réelle : 0.5 s
  Position détectée : 0.51 s
  Détection correcte : 1
  Norme de Frobenius min : 80.8808
  Corrélation : 0.97519
Prononciation de D
  Position réelle : 0.7 s
  Position détectée : 0.71 s
  Détection correcte : 1
  Norme de Frobenius min : 79.5843
  Corrélation : 0.9756
```



Question9:

Question 9 : Dcase 2025 challenge: task 3 : Stero sound event localization and detection (SELD) in regular video content:

voici le lien de DCASE [DCASE2025 Challenge - DCASE]

voici le lien de task3: Stereo sound event localization and detection in regular video content - DCASE

on a choisi la Task 3 du challenge DCASE 2025, intitulée "sound event localization and detection (SELD) in regular video content ", car elle a pour objectif principal de détecter et localiser des sons dits événements dans du contenu vidéo régulier (comme des vidéos de sport, films, documentaires, etc.) ce qui revient à faire du " Template Matching" pour la reconnaissance d'events . (exemple: détecter le mot "goal" lors d'un match de football / reconnaître certains mots qui peuvent être à caractèredérogatoire pour prévenir l'audience ...)

Objectif de la Task 3 (SELD in Regular Video Content)

L'objectif est de développer un système capable d'identifier quels événements sonores se produisent, quand ils se produisent (détection temporelle), où ils se produisent (localisation spatiale), et s'ils sont actifs ou non à un moment donné.

En clair, pour chaque son dans la vidéo (ex. applaudissements, voix, sirène, coup de feu), le système doit :
● Détecter le type de son et Préciser l'instant où il commence et se termine.
● Estimer la direction d'où le son provient (azimut et élévation).
● Gérer la présence de plusieurs sons en même temps (événements simultanés).

Applications d'un tel algorithme
Voici quelques cas d'usage concrets :
● Analyse de contenu vidéo : pour automatiser le sous-titrage enrichi avec des annotations sonores ("un chien aboie à gauche", "sirène de police à droite").
● Robotique et IA multimodale : pour permettre à un robot de comprendre son environnement sonore et réagir (ex. tourner vers une personne qui parle).
● Surveillance intelligente : détection de sons suspects avec leur position (ex. un bruit de bris de verre à droite de la caméra).
● Aide aux malvoyants : offrir un retour audio intelligent sur les sons dans une vidéo ou l'environnement.
● Jeux vidéo / Réalité augmentée : immersion réaliste basée sur l'orientation et la source sonore.

● Indexation de contenu multimédia : classer automatiquement les vidéos par types de sons présents.

Sound event classes:

1. Female speech, woman speaking
2. Male speech, man speaking
3. Clapping
4. Téléphone
5. Laughter
6. Domestic sounds
7. Walk, footsteps
8. Door, open or close
9. Music
10. Musical instrument
11. Water tap, faucet
12. Bell
13. Knock

Méthodes et approches courantes

● Modèles CNN-RNN ou Transformers pour traiter l'audio spatial.
● Utilisation de CRNN (Convolutional Recurrent Neural Networks) pour capturer les dépendances temporelles et spatiales.
● Représentations audio + “ Template Matching” : spectrogrammes log-mel, interaural level/time differences, intensité vectorielle.

Évaluation

Les systèmes sont évalués avec deux métriques principales :

1. Event Detection :
   ○ Précision, rappel, F-score (combien de bons sons sont détectés au bon moment).

2. Localization Accuracy :
   - Erreur de direction (distance angulaire entre la prédiction et la vérité).
   - SELD score combiné : intégrant à la fois la détection et la localisation.