Université Sidi Mohamed Ben Abdellah
Faculté des sciences Dhar El Mahraz

Année universitaire : 2024/2025
Master MIDVI

TP3

Activité 1

Dans cette activité, nous allons réaliser une détection de visage dans une vidéo à l'aide de **MTCNN** et une reconnaissance de visage avec **LBP (Local Binary Patterns)** et **SVM (Support Vector Machine)**, nous allons suivre les étapes suivantes :

1. **Détecter les visages** : Appliquer MTCNN pour détecter les visages dans chaque frame de la vidéo.
2. **Extraire les caractéristiques LBP** : Pour chaque visage détecté, extraire les descripteurs LBP. On convertit chaque visage détecté en niveaux de gris et extrait le descripteur LBP, puis on le normalise.
3. **Reconnaître les visages** : Utiliser un modèle SVM entraîné pour prédire l'identité des visages. Pour cela, on va créer une fonction qui charge un modèle SVM préexistant à partir d'un fichier. Si le modèle n'existe pas, on entraîne un nouveau modèle à partir des données d'entraînement que vous devez fournir.

Prérequis

```
pip install tensorflow keras opencv-python scikit-learn
```

Remarques

- **Données d'Entraînement** : Assurez-vous d'entraîner le modèle SVM avec suffisamment de données pour chaque personne. Vous pouvez créer un ensemble de données d'images de visages pour cela.
- **Performance** : La vitesse de traitement dépend de la taille de l'image, du nombre de visages détectés et de la complexité du modèle SVM.

```
# Fonction de détection de visages
def detect_faces(image):
    detector = MTCNN()
    faces = detector.detect_faces(image)
    return faces
```

```
# Fonction pour extraire des caractéristiques LBP
def extract_lbp(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = feature.local_binary_pattern(gray, P=8, R=1, method="uniform")
```

```
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, 11), range=(0, 10))
    hist = hist.astype("float")
    hist /= hist.sum()
    return hist
```

```
# Fonction pour entraîner le modèle SVM
def train_svm(features, labels):
    model = svm.SVC(kernel='linear', probability=True)
    model.fit(features, labels)
    return model
```

```
# Charger ou entraîner le modèle SVM
def load_or_train_svm(training_data, training_labels):
    # Vérifier si le modèle existe
    try:
        model = joblib.load("face_recognition_model.pkl")
    except FileNotFoundError:
        model = train_svm(training_data, training_labels)
        joblib.dump(model, "face_recognition_model.pkl")
    return model
```

```
# Charger la vidéo
cap = cv2.VideoCapture(0)
# Liste pour stocker les caractéristiques et les étiquettes
features = []
labels = []
# Entraîner le modèle SVM avec les données d'entraînement
model = load_or_train_svm(features, labels)
# Boucle de traitement de la vidéo
while True:
```

```
    ret, frame = cap.read()
    if not ret:
        break
    # Détecter les visages
    faces = detect_faces(frame)
    for face in faces:
        x, y, width, height = face['box']
        detected_face = frame[y:y+height, x:x+width]
        # Extraire LBP pour le visage détecté
        lbp_features = extract_lbp(detected_face)
        # Prédiction avec le modèle SVM
        prediction = model.predict([lbp_features])
        # Affichage des résultats
        cv2.rectangle(frame, (x, y), (x + width, y + height), (0, 255, 0), 2)
        cv2.putText(frame, prediction[0], (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,
0), 2)
    # Afficher le cadre avec les détections
    cv2.imshow("Détection et Reconnaissance de Visage", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Activité 2

Dans cette activité, nous allons explorer un autre vecteur descripteur **HOG (Histogram of Oriented Gradients) et SIFT(Scale-Invariant Feature Transform)**. Nous gardons toujours le même classifieur SVM pour comparer les performances de chaque descripteur.