

פרויקט HAND SKELETON

תיאור האפליקציה

הפרויקט הוא מערכת לזיהוי והצגה בזמן אמת של שלד יד בתלת-ממד מתוך וידאו. המערכת משלבת בין Python ל-C++, כאשר רכיב ה-Python משתמש ב-MediaPipe של Google לזיהוי 21 נקודות שלד (Landmarks) בכל יד, ורכיב ה-C++ משתמש ב-OpenGL להצגת השלד בתלת-ממד. שני הרכיבים פועלים במקביל ומתקשרים ביניהם באמצעות קובץ JSON שמתעדכן בכל פריים. ממשק גרפי (GUI) ב-Tkinter מאפשר למשתמש לבחור וידאו, להפעיל ולעצור את המערכת, ולהשהות את העיבוד בזמן אמת. המערכת תומכת בזיהוי של עד שתי ידיים במקביל, ומציגה את המפרקים וה"עצמות" בצורה ויזואלית בתלת-ממד.

טכנולוגיות:

- MediaPipe - זיהוי של 21 נקודות שלד (Landmarks) בכל יד.
- OpenCV (opencv-python) - קריאת וידאו, עיבוד פריימים, והצגת התמונה המקורית.
- Tkinter - ממשק גרפי להפעלת המערכת, בחירת וידאו, שליטה בהשהיה/עצירה. NumPy - חישובים ופעולות עזר (תלות עקיפה של MediaPipe ו-OpenCV).
- OpenGL C++ - רינדור גרפי בתלת-ממד של שלד היד.
- GLU / GLUT (FreeGLUT) - ניהול חלון, אירועים ותצוגה.
- nlohmann/json - ספרייה לקריאת קובץ JSON עם נתוני הנקודות.
- C++17 - שפת התכנות לרכיב ההצגה. מערכת וכלים משלימים
- Makefile - בנייה קומפילציה של רכיב ה-C++.
- קובצי Flag (pause.flag / done.flag) - שליטה וסנכרון בין הרכיבים.
- מבנה JSON - קובץ current.json לשיתוף המידע בין Python ל-C++.

מבנה

```
3D-Hand-Skeleton/
|
├── assets/                                # תיקיית משאבים וקבצי עבודה
|   ├── hand_video.mp4                    # קובץ הווידאו הנבחר לעיבוד
|   ├── current.json                      # עם נקודות היד (מתעדכן כל פריים) JSON קובץ
|   ├── pause.flag                        # (Pause) דגל להשהיית העיבוד
|   ├── done.flag                         # (End of video) דגל לסיום העיבוד
|   └── mano_models/                     # (לא בשימוש כרגע) MANO קבצי מודל
|       └── קבצי וידאו/תמונות לדוגמה
|
├── python/                               # קוד Python
|   ├── gui_launcher.py                  # ממשק גרפי להפעלה, השהיה ועצירה
|   └── track_video.py                   # JSON מעקב אחר הידיים וכתובת נקודות ל-
|
├── src/                                  # C++ קוד מקור
|   ├── main.cpp                         # של שלד היד OpenGL רינדור
|   └── hand_loader.cpp                  # C++ והמרתו למבנה ב- JSON קריאת קובץ
|
├── include/                             # C++ של Header קבצי
|   └── hand_loader.h                   # ופונקציית טעינת הנקודות Landmark הגדרת מבנה
|
├── build/                               # תיקיית פלט של הקומפילציה
|   └── hand_skeleton                   # C++ קובץ ההרצה של רכיב ה-
|
├── Makefile                             # C++ והוראות בנייה של רכיב ה-
└── README.md                           # תיעוד והוראות שימוש
```

פירוט קבצים:

• **gui_launcher.py**

תפקיד: ממשק גרפי (GUI) ב-Tkinter שמנהל את הפעלת המערכת.

פעולות עיקריות:

בחירת קובץ וידאו והעתקתו ל-assets/hand_video.mp4.

הרצה במקביל של: track_video.py (מעקב MediaPipe).
build/hand_skeleton (מציג ה-C++).

השהייה/המשך באמצעות יצירה ומחיקה של pause.flag.

עצירה וסגירה של שני התהליכים יחד.

ניטור מצב התהליכים - אם אחד נסגר, סוגר גם את השני.

• **track_video.py**

תפקיד: מודול מעקב אחר נקודות הידיים בוידאו.

פעולות עיקריות:

פתיחת וידאו מ-assets/hand_video.mp4 עם OpenCV. שימוש ב-MediaPipe Hands לזיהוי עד שתי ידיים (21 נקודות לכל יד).

כתיבת הנתונים (x, y, z) ל-assets/current.json בפורמט JSON, בעדכון אטומי (temp file → החלפה).

בדיקת קובץ pause.flag - אם קיים, עוצר זמנית את העיבוד.

הצגת הווידאו בחלון OpenCV בזמן ריצה.

בסיום הסרטון: סוגר את הווידאו, מוחק חלונות, וכותב done.flag.

• main.cpp

תפקיד: מציג גרפי ב-OpenGL שמצייר את שלד היד בתלת-ממד על בסיס קובץ JSON.

פעולות עיקריות:

קריאה ל-load_hand_landmarks() כדי לטעון את assets/current.json שמירת הנקודות במבנה <<std::vector<std::vector<Landmark

וקטור של ידיים, כל יד עם 21 נקודות.

ציור: נקודות המפרקים בירוק (GL_POINTS).

קווים לבנים בין נקודות לפי מבנה העצמות של MediaPipe (GL_LINES).

עדכון התצוגה כל 33ms (~30FPS).

בדיקה אם done.flag קיים - במידה וכן, סוגר את התוכנית.

• hand_loader.cpp

תפקיד: קריאת קובץ JSON והמרתו למבנה נתונים ++C.

פעולות עיקריות:

פתיחת הקובץ בעזרת std::ifstream.

שימוש בספריית nlohmann/json כדי לפרסר את המידע.

מעבר על כל הידיים והנקודות, בניית Landmark לכל נקודה (x, y, z).

החזרת מבנה <<std::vector<std::vector<Landmark לשימוש במציג.

ספריות שיש להתקין:

להורדת mediapipe, opencv, opengl דרוש להריץ:

```
sudo apt update
```

```
sudo apt install build-essential libgl1-mesa-dev libglu1-mesa-dev  
  \ freeglut3-dev
```

```
nlohmann-json3-dev python3-pip python3-tk ffmpeg
```

```
pip3 install mediapipe opencv-python
```

הרצה:

לקמפל בעזרת ה makefile את הקוד:

```
make clean && make
```

להריץ את הפקודה:

```
python3 python/gui_launcher.py
```

לבחור סרטון שרוצים מהGUI

ללחוץ על כפתור הSTART