דו״ח טכני – ROBO FRIEND

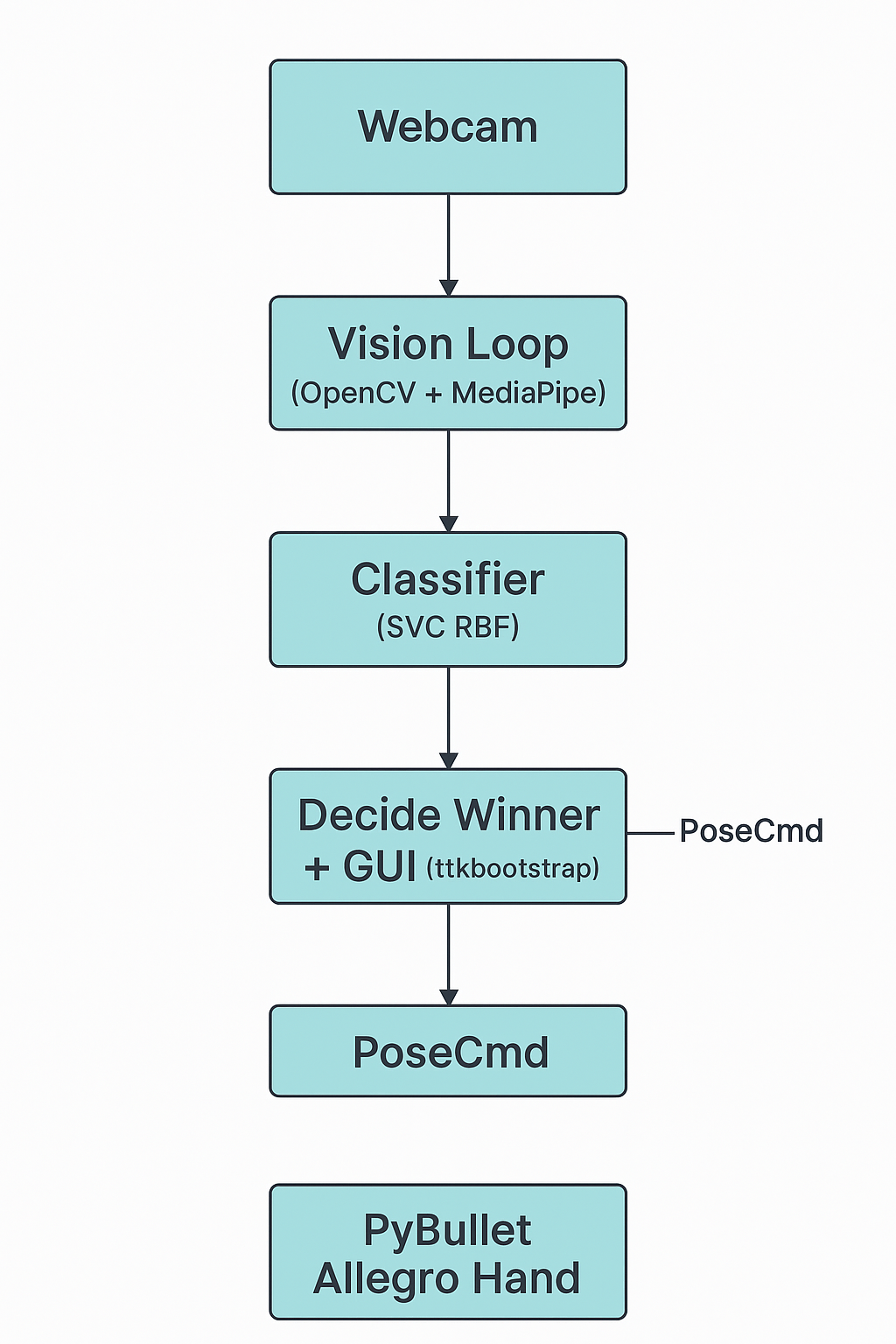
מבוא:

מטרת הפרויקט הייתה לבנות מערכת אינטראקטיבית המדמה משחק “אבן‑נייר‑מספריים” בין משתמש אנושי לבין יד רובוטית וירטואלית (Allegro Hand). החזון: המשתמש מציג מחווה מול מצלמת‑רשת, האלגוריתם מזהה בזמן‑אמת את המחווה, בוחר עבור הבוט תשובה אקראית, מציג את היד הרובוטית בסביבת PyBullet ומעדכן ניקוד.

ארכיטקטורת מערכת:

| **שכבה** | **ספריות** | **אחריות עיקרית** |
| --- | --- | --- |
| GUI Launcher | ttkbootstrap | בחירת מספר סיבובים, חלון עזרה, ניהול מצבים (WAIT/COUNT/SHOW/DONE) |
| Vision Loop | OpenCV, MediaPipe Hands | לכידת הפריים, הפקת ‏21‑Landmarks |
| Classifier | scikit‑learn + joblib | סיווג המחווה ל‑Rock/Paper/Scissors |
| Physics | PyBullet, allegro\_hand\_description | אנימציית כף‑היד במקביל ללולאת ה‑GUI |
| Utility | NumPy, JSON | אחסון הגדרות, חישובים גאומטריים |

תרשים:



**איך לימדנו את המחשב לזהות אבן‑נייר‑מספריים – בקצרה ובפשטות**

1. **איסוף דוגמאות**  
   הפעלנו סקריפט שמראה בקונסולה איזו מחווה להציג. הצבנו את היד מול המצלמה, וכל פעם שעשינו Rock, Paper או Scissors הסקריפט שמר קובץ קטן (‎JSON) עם 21 נקודות‑מפתח של היד מה‑ MediaPipe ועוד תגית של המחווה. אספנו בערך 200 תמונות לכל מחווה – ביחד ≈ 600 דוגמאות.
2. **סידור הנתונים**  
   קבצי  JSON הפכנו לטבלה אחת: בכל שורה 63 מספרים - x,y,z של כל 21 הנקודות. לפני האימון נירמלנו – מחסירים את שורש כף‑היד ומחלקים באורך כף‑היד – כך שהגדלים והמרחק מהמצלמה לא משפיעים.
3. **חלקים ל‑אימון/בדיקה**  
   חילקנו 80 % מהשורות כדי ללמד (Train) ו‑20 % כדי לבדוק (Test). עשינו זאת “בשכבות” כך שכל מחווה תישאר בפרופורציה.
4. **אימון המודל**  
   בחרנו מסווג SVM עם גרעין RBF. אחרי ניסוי כמה ערכים קבועים בחרנו ‎C = 5, γ = 0.8.  
   התוצאה – כשהרצנו על סט הבדיקה קיבלנו דיוק ~ 96 %.

**בזמן‑ריצה**  
בכל פריים המצלמה, אנחנו:

* 1. מוציאים 21 Landmarks.
  2. עושים לה נורמליזציה.
  3. שולחים למודל → הוא מחזיר Rock / Paper / Scissors.  
     אם הוא לא בטוח הסתברות נבחרת מחווה אקראית כדי לא לעכב את המשחק.

למה זה טוב? לא דרוש GPU, הקובץ קטן, והזיהוי רץ חלק.