Univerza v Ljubljani
Fakulteta za elektrotehniko

Univerza v Ljubljani

Fakulteta za elektrotehniko

Robotski vid

Seminarska naloga - Izziv

Poročilo

Študent: Alexey Yudin

Maj 2025

Laboratorij za
Slikovne tehnologije
http://lit.fe.uni-lj.si

Kazalo

Povzetek 3
Uvod 3
Metodologija 4
Rezultati 7
Diskusija 8
Priloga 1. Struktura podatkov projekta 9
Priloga 2. Opis skript projekta 10
- **validate_single_video.py** 10
- **visualize_combined_dataset_with_predictions.py** 10
- **extract_keypoints.py** 12
- **merge_labels.py** 12
- **visualize_keypoints.py** 12
- **train_combined_dataset.py** 12

Povzetek

V okviru tega projekta je bil razvit algoritem za samodejno prepoznavanje faz pri izvajanju testa devetih zatičev (9HPT) z uporabo metod računalniškega vida. S pomočjo knjižnice MediaPipe so bili iz video posnetkov izluščeni koordinati 21 ključnih točk roke, ki so nato služili za učenje klasifikacijskega modela (Random Forest). Model ločuje glavne faze testa: prijem zatiča, prenos, vstavljanje in gibanje s prazno roko. Za interpretacijo rezultatov so bile implementirane vizualizacije, ki prikazujejo predvidene faze in verjetnosti neposredno na posnetku. Sistem podpira tudi samodejno generiranje označenih datotek in kvantitativno validacijo napovedi v primerjavi z ročno označenimi podatki. Posebna pozornost je bila namenjena robustnosti klasifikacije in delovanju ob prisotnosti šuma v podatkih. Raziskane so bile tudi različne metode za izboljšanje kakovosti prepoznavanja, usmerjene v odpravo tipičnih napak modela in povečanje zanesljivosti klasifikacije.

Uvod

Test devetih zatičev (9HPT, Nine-Hole Peg Test) je standardna metoda za ocenjevanje motoričnih in koordinacijskih sposobnosti roke, ki se pogosto uporablja v klinični praksi in rehabilitaciji. Tradicionalno se ocena testa izvaja ročno, kar zahteva prisotnost strokovnjaka in je podvrženo subjektivnim dejavnikom. Avtomatizacija analize izvajanja testa 9HPT lahko bistveno poveča objektivnost in učinkovitost spremljanja bolnikovega stanja.

Med možnimi rešitvami je posebno zanimiva uporaba metod računalniškega vida. Sodobna orodja, kot je knjižnica MediaPipe, omogočajo pridobivanje ključnih točk roke iz video posnetkov v realnem času, kar odpira možnosti za analizo faz testa na podlagi gibanja in položajev prstov. Vendar pa naloga razvrščanja faz zahteva odpornost na šum in variabilnost gibov med uporabniki.

Cilj tega projekta je bil ustvariti enostavno in razširljivo orodje, ki lahko samodejno prepozna ključne faze izvajanja testa 9HPT na podlagi koordinat ključnih točk roke. Predlagani pristop ne zahteva specializirane opreme in se lahko izvaja na osnovi običajnih video posnetkov. V okviru projekta so bila razvita orodja za obdelavo podatkov, učenje modela, vizualizacijo napovedi in validacijo rezultatov.

Metodologija

Postopek izgradnje sistema za samodejno prepoznavanje faz je vključeval več zaporednih korakov:

1. Izluščenje značilk
Za pridobivanje koordinat ključnih točk roke je bila uporabljena knjižnica MediaPipe Hands. Iz vsakega videa so bile za vsak okvir izluščene koordinate 21 točke (x, y, z). Obdelavo je izvajal skript extract_keypoints.py, rezultati pa so bili shranjeni v obliki CSV.

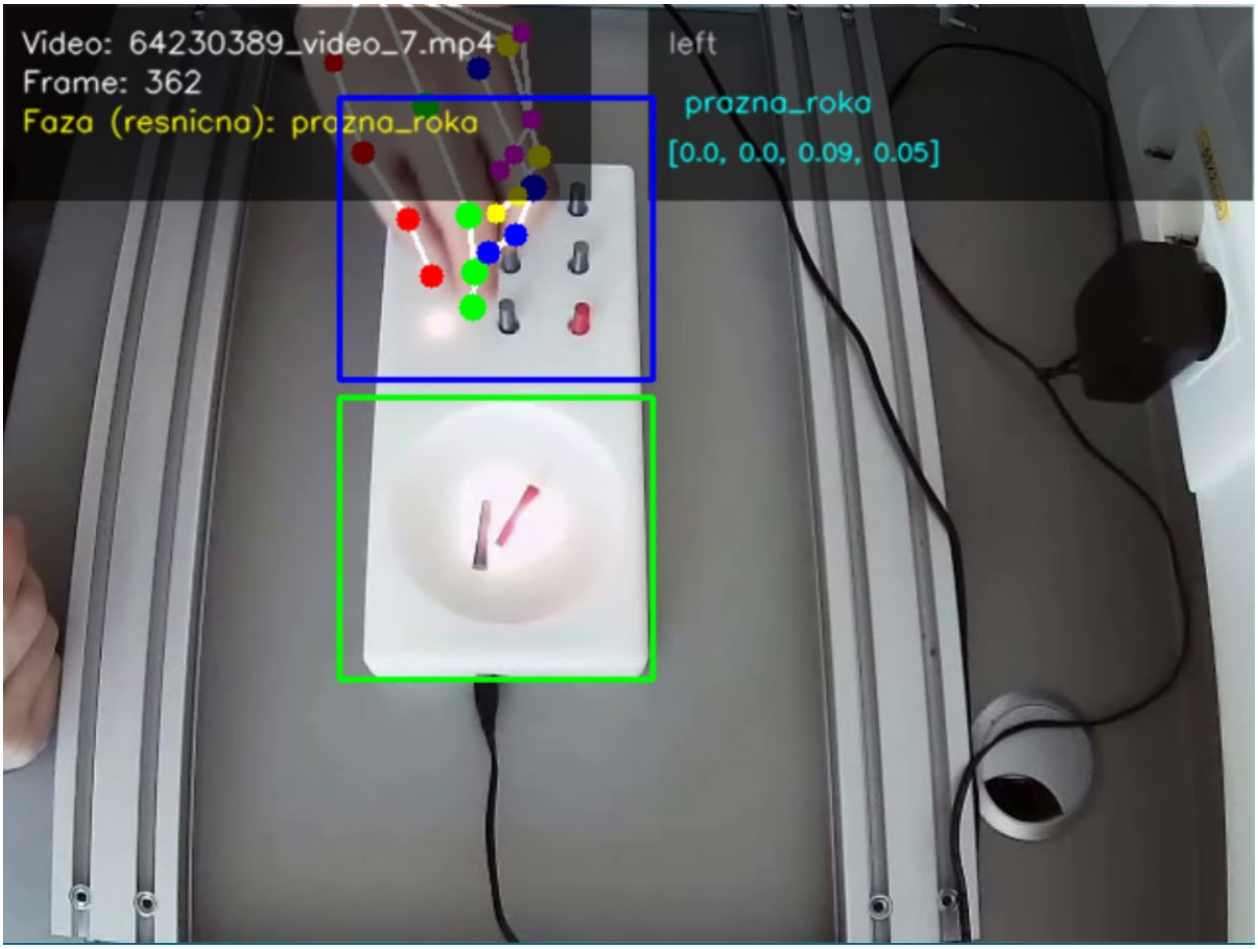
2. Anotacija in priprava podatkovnega nabora
Skript merge_labels.py združi koordinate točk z fazami iz ročno označenih podatkov (JSON). Vsakemu okviru je dodeljena ustrezna faza: prazna roka, prijem zatiča, prenos, vstavljanje. Vsi označeni podatki so bili združeni v datoteko combined_dataset.csv. Ročno označeni JSON-fajli so bili povezani s koordinatami točk, pri čemer so bili odstranjeni vsi okviri z ničelnimi vrednostmi.

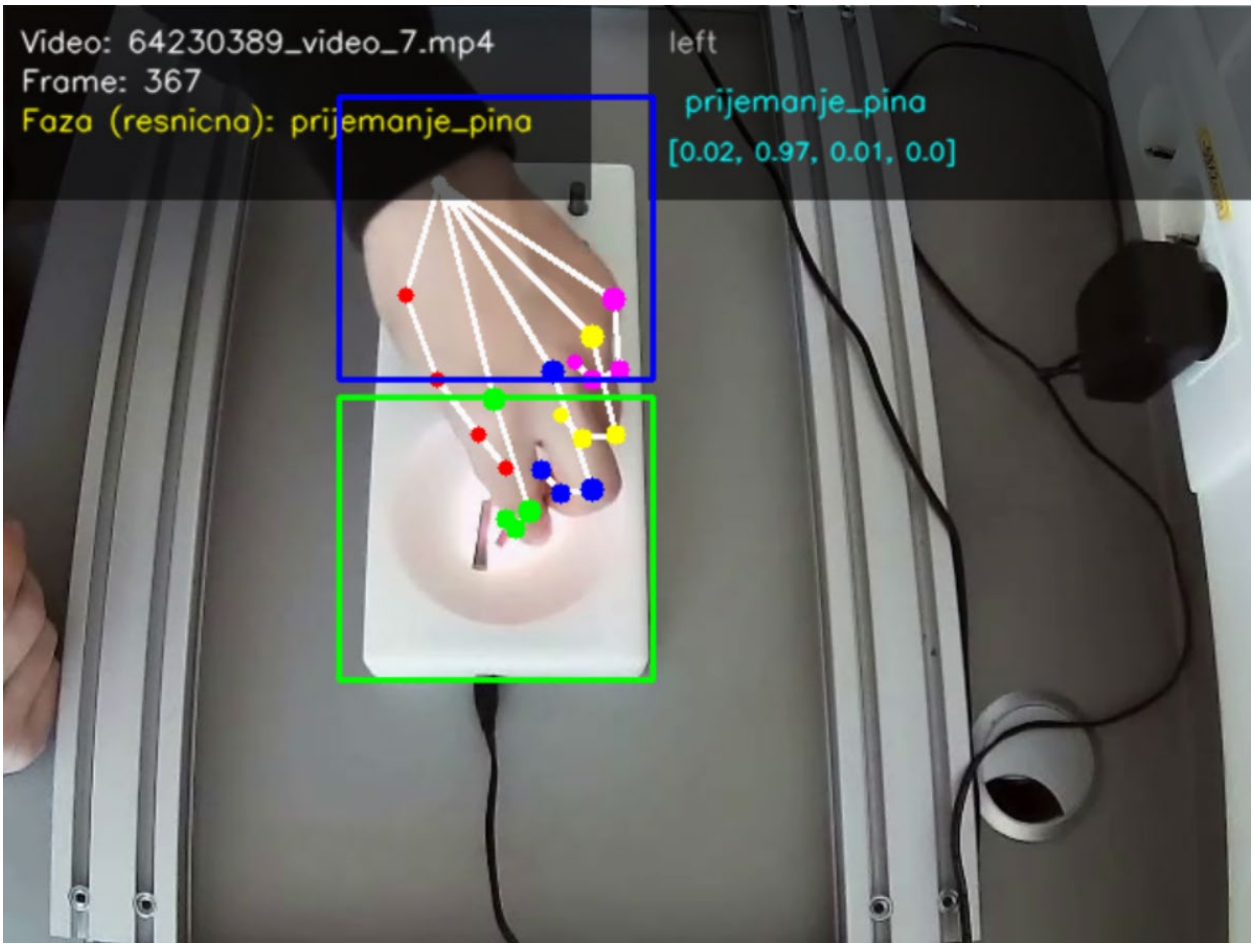
3. Učenje modela
Na podlagi datoteke combined_dataset.csv je bil za klasifikacijo faz izurjen model Random Forest iz knjižnice scikit-learn (skript train_combined_dataset.py). Vhodne značilke so bile koordinate točk, ciljne oznake pa faze. Naučeni model je bil shranjen kot trained_model.pkl. Od skupno 2688 vrstic je bilo 2200 uporabljenih za učenje, preostale pa za testiranje.

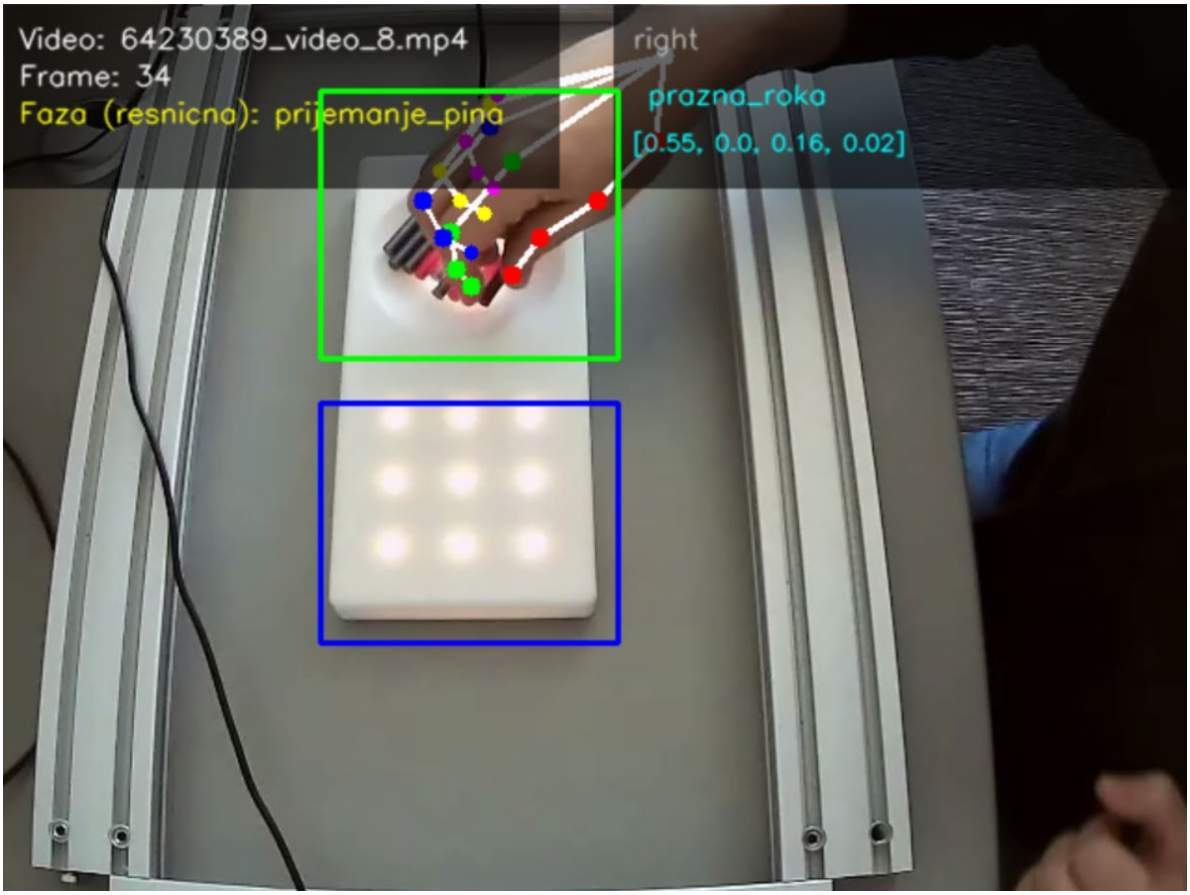
4. Vizualizacija in preverjanje
Za vizualno preverjanje delovanja modela je bil implementiran način vizualizacije, ki prikazuje ključne točke, faze in verjetnosti razvrščanja na video. Na voljo je bil tudi hiter način brez prikaza videa, ki prikazuje le metrike natančnosti. Skript visualize_combined_dataset_with_predictions.py izvaja glavni algoritem prepoznavanja v treh načinih:
• hitra analiza CSV;
• vizualizacija napovedi na videu;
• kombinirana analiza videa in CSV.

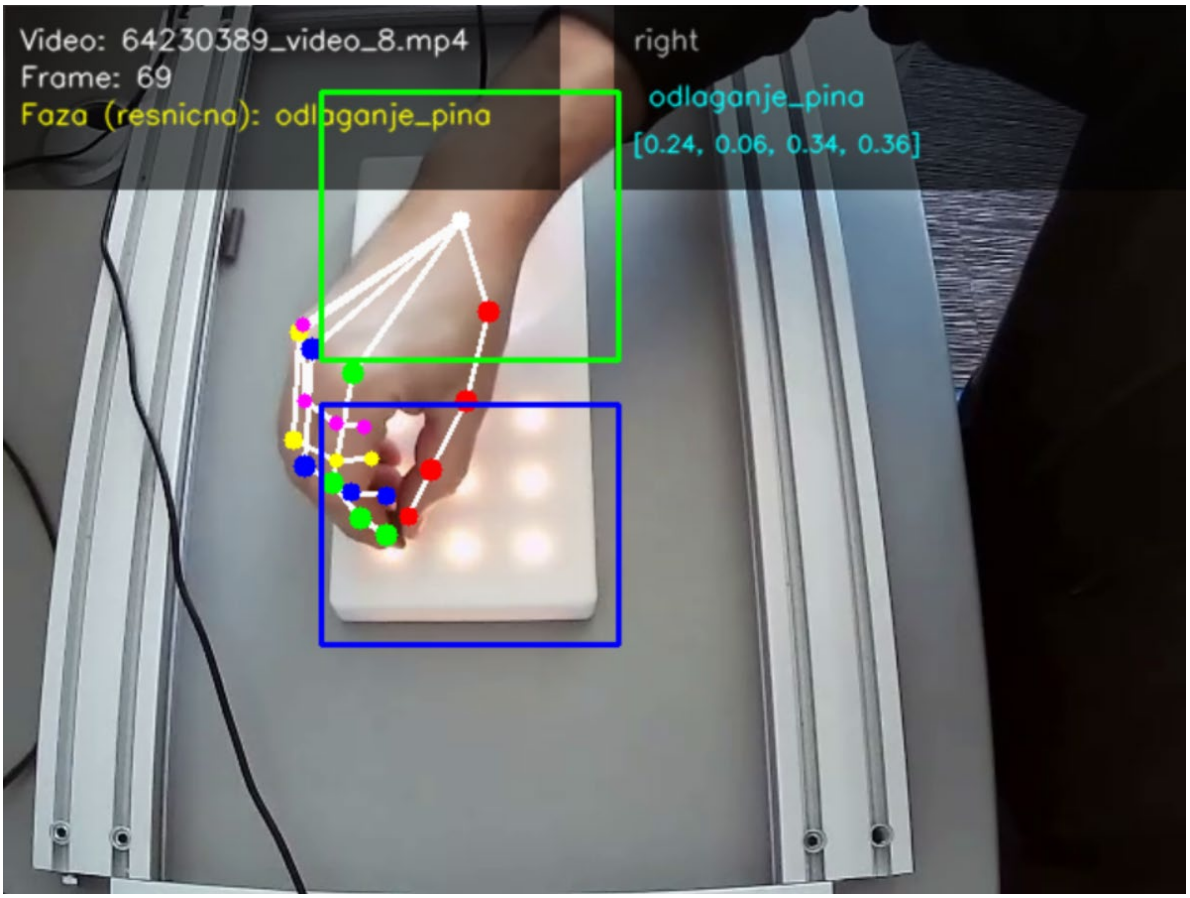
Po začetni klasifikaciji model dodatno obdela rezultate z naslednjimi heuristikami:
• glajenje z uporabo stabilnih zaporedij (funkcija fix_by_stable_sequence);
• izločanje faz v območjih brez gibanja (funkcija no_movement_zones);
• samodejna določitev strani roke z uporabo referenčnih slik (funkcija detect_hand_side).
Vse ključne točke, faze, verjetnosti in delovna območja so prikazana neposredno na kadrih videa.

5. Validacija in izboljšanje napovedi
Raziskane so bile različne metode za izboljšanje natančnosti razvrščanja, vključno s:
• korekcijo na podlagi stabilnih zaporedij okvirjev z visoko verjetnostjo iz Random Foresta,
• izločanjem lažnih prepoznav v območjih brez gibanja,
• uporabo značilnih razporeditev točk za posamezne faze.


Video: 64230389_video_7.mp4
Frame: 362
Faza (resnicna): prazna_roka
left
prazna_roka
[0.0, 0.0, 0.09, 0.05]


Video: 64230389_video_7.mp4
Frame: 367
Faza (resnicna): prijemanje_pina
left
prijemanje_pina
[0.02, 0.97, 0.01, 0.0]


Video: 64230389_video_8.mp4
Frame: 34
Faza (resnicna): prijemanje_pina
right
prazna_roka
[0.55, 0.0, 0.16, 0.02]


Video: 64230389_video_8.mp4
Frame: 69
Faza (resnicna): odlaganje_pina
right
odlaganje_pina
[0.24, 0.06, 0.34, 0.36]

Rezultati

Na žalost so bili rezultati nezadovoljivi. V najboljšem primeru je bilo mogoče doseči pravilno prepoznavanje faz v 60 %, pri čemer so bili rezultati po posameznih fazah naslednji:

Osnovni rezultat je zagotavljal model Random Forest

=== Rezultat po razredih ===

odlaganje_pina : 34 / 89 (38.2%)

prazna_roka : 87 / 127 (68.5%)

prenos_pina : 103 / 138 (74.6%)

prijemanje_pina : 57 / 125 (45.6%)

SKUPAJ : 281 / 479 (58.7%)

Kot rezultat postopne obdelave z metodama fix_by_stable_sequence in no_movement_zones so bila izboljšanja minimalna.

=== Rezultat po razredih ===

odlaganje_pina : 34 / 89 (38.2%)

prazna_roka : 89 / 127 (70.1%)

prenos_pina : 103 / 138 (74.6%)

prijemanje_pina : 57 / 125 (45.6%)

SKUPAJ : 283 / 479 (59.1%)

Diskusija

Pridobljeni rezultati kažejo, da ima predlagani pristop v trenutni obliki omejeno natančnost in trenutno še ne more služiti kot zanesljiva rešitev za praktično uporabo. Čeprav je uporaba knjižnice MediaPipe omogočila učinkovito izluščenje koordinat ključnih točk roke, se je nadaljnja klasifikacija faz izkazala za zahtevno nalogo.

Kot je razvidno iz poglavja Rezultati, je osnovni model Random Forest dosegel skupno natančnost 58,7 %. Pri tem obstajajo velike razlike med posameznimi fazami. Faza prenosa zatiča (prenos_pina) je dosegla 74,6 %, medtem ko sta bili fazi prijema (prijemanje_pina) in odlaganja (odlaganje_pina) pod 50 %. To je verjetno posledica visoke variabilnosti položaja roke in kratkega trajanja teh faz.

Implementirani so bili tudi nekateri postopki naknadne obdelave napovedi, namenjeni odpravljanju značilnih napak: glajenje stabilnih zaporedij (fix_by_stable_sequence) in zatiranje napačnih faz v območjih brez gibanja (no_movement_zones). Vendar pa je bil njihov vpliv na končno natančnost majhen – skupna natančnost se je izboljšala le za 0,4 %.

Analiza video gradiva je pokazala, da so nekatere faze težko razločljive tudi vizualno, zlasti pri hitrih ali netipičnih gibanjih. Poleg tega je bil model učen na omejenem številu podatkov (le 2200 vrstic za učenje), pridobljenih iz majhnega števila posnetkov, kar je negativno vplivalo na njegovo sposobnost posploševanja.

Izvorna ideja avtorja projekta je bila zasnova dvostopenjskega postopka prepoznavanja. V prvi fazi naj bi model Random Forest podal začetne verjetnosti za posamezne razrede, v drugi fazi pa bi zaporedni moduli s posebnimi heuristikami postopoma popravljali napovedi. Vsak tak modul bi bil namenjen reševanju konkretne naloge – na primer, izločitev nemogočih faz ob odsotnosti ključnih točk v določenem območju ali odstranitev faze prenosa ob odsotnosti usmerjenega gibanja.

Načrtovana je bila izgradnja celotnega procesnega cevovoda z okoli 10 takimi moduli, ki bi postopoma izboljševali rezultate in odpravljali tipične napake modela. Žal je bilo v okviru tega projekta mogoče implementirati le dva od teh postopkov. Kljub temu je bila arhitektura zasnovana kot razširljiva, kar omogoča nadaljnji razvoj in izboljšave v prihodnosti.

Priloga 1. Struktura podatkov projekta

Mapa vhodnih podatkov Input:

- 8 videodatotek (2 datoteki od 10 so bile poškodovane med snemanjem)
- 8 označenih datotek .json *(ročno označene faze testa)*
- combined_dataset.csv - tabela koordinat ključnih točk dlani, sestavljena na podlagi 8 videodatotek in 8 .json datotek in vsebuje samo prepoznane okvire (vrstice z ničelnimi vrednostmi so bile odstranjene)

  Značilnosti:

  2688 vrstic

  Vsaka vrstica vsebuje:

  63 vrednosti (x, y, z za 21 točk)

  številko okvirja

  ime datoteke

  oznako faze

- trained_model.pkl, trained_model2.pkl
  Naučeni modeli za klasifikacijo faz, ustvarjeni z algoritmom Random Forest

- left_reference.png, right_reference.png
  Referenčni okvirji za levo in desno roko, uporabljeni za določitev delovnih območij v videu

Mapa izhodnih podatkov Output:

Namenjena shranjevanju .json datotek z referenčnimi oznakami (ground truth) za testiranje.

Priloga 2. Opis skript projekta

validate_single_video.py

Primer ukaza za zagon:
python validate_single_video.py Input/64230389_video_1.mp4 Input/64230389_video_1_sk.json Output/64230389_video_1_output.json

visualize_combined_dataset_with_predictions.py

Primer ukaza za zagon:
python visualize_combined_dataset_with_predictions.py (use_video_input = False)

Vhodni podatki:

- Input/combined_dataset.csv – tabela s koordinatami 21 ključne točke roke in oznakami faz
- Input/trained_model.pkl – naučen model (Random Forest)
- Input/*.mp4 – videodatoteka (če je aktiviran način z videom)
- Input/left_reference.png, Input/right_reference.png – referenčni okvirji za funkcijo detect_hand_side(frame, left_ref, right_ref)

Izhodni podatki:

- Vizualizacija s prikazom predvidenih in dejanskih faz, ključnih točk, verjetnosti
- .json datoteka z oznakami faz po okvirjih (če je zagnano z videom)
- Izpis metrik v konzoli (v hitrem načinu)

Načini delovanja:

1. **Hiter način (quick test mode)**
   – Uporabi se le combined_dataset.csv, izračunajo se verjetnosti in metrike
   – Preprosta heuristika določi stran roke
2. **Vizualizacija z videom**
   – Iz videa se izluščijo ključne točke, izvede se klasifikacija faz, izrišejo se rezultati
3. **Vizualizacija z videom in CSV**
   – Uporabijo se obstoječe točke iz combined_dataset.csv in ustrezni kadri iz videa

Opis logike delovanja:

- Naloži se model
- Če je use_video_input = True, se obdeluje video in ustvari .json z napovedmi
- Če ne, se obdeluje combined_dataset.csv
- Po klasifikaciji z Random Forest se rezultati dodatno obdelajo:
  - stabilizacija faz (funkcija fix_by_stable_sequence)
  - zatiranje faz v območjih brez gibanja (no_movement_zones)
- Na zaslon se izpišejo: številke okvirjev, ključne točke, območja, faze, verjetnosti

Opis metod skripte
visualize_combined_dataset_with_predictions.py

load_model(path)	Naloži model iz .pkl datoteke.
load_dataset()	Naloži CSV s koordinatami in fazami.
extract_keypoints_from_video(video_path)	Izlušči ključne točke iz videa z uporabo MediaPipe.
process_video_input(video_path, model, output_json_path=None)	Obdelava videa: izluščenje, napoved, korekcija, vizualizacija.
build_hand_array(df, frames_by_video=None)	Določi stran roke na podlagi koordinat ali slike.
correct_probabilities(y_proba_array, df=None, hand_array=None)	Korekcija verjetnosti na podlagi stabilnosti in odsotnosti gibanja.
fix_by_stable_sequence(...)	Odstrani šum v napovedih na podlagi stabilnih faz.
no_movement_zones(...)	Nastavi verjetnost na nič v območjih brez gibanja.
draw_keypoints(frame, row)	Nariše ključne točke in povezave na kadru.
draw_zones(frame, hand)	Nariše območja na kadru.
draw_overlay(...)	Prikaže besedilo: fazo, roko, verjetnosti itd.
play_frames(...)	Predvaja video po kadrih z možnostjo nadzora.
load_video_file(path)	Naloži video in preveri dostopnost.
load_frames(video_frames_map)	Naloži zahtevane kadre v pomnilnik.
collect_frames_to_load(df)	Ustvari seznam potrebnih kadrov.
run_quick_test(df, model, y_proba_array)	Hiter način za oceno modela.
save_predictions_as_json(predicted_labels, output_path)	Shrani napovedi po kadrih v JSON.
detect_hand_side(frame, left_ref, right_ref)	Določi stran kamere

extract_keypoints.py

Izluščenje ključnih točk dlani iz videa z uporabo knjižnice MediaPipe in shranjevanje rezultata v CSV datoteko. Vsaka vrstica v datoteki ustreza enemu kadru.

Glavne funkcije:

- Obdelava videa kader za kadrom
- Detekcija 21 ključne točke roke (x, y, z)
- Shranjevanje koordinat v CSV (ena datoteka na video)

merge_labels.py

Združevanje koordinat ključnih točk (CSV) in označenih dogodkov (JSON) za en video. Rezultat je CSV-datoteka, kjer je vsakemu kadru (vrstici) dodeljena ustrezna oznaka dejanja (faza izvajanja testa).

Glavne funkcije:

- Nalaganje koordinat iz CSV
- Nalaganje anotacij iz JSON
- Dodelitev oznake vsakemu kadru na podlagi obsegov okvirjev
- Shranjevanje končne datoteke z oznakami

visualize_keypoints.py

Interaktivni ogled videa z naloženimi ključnimi točkami in fazami. Uporabno za preverjanje izluščenih značilk in kakovosti anotacij. Vizualizira ključne točke roke, prikazane na videu skupaj s fazo dejanja, številko kadra in časom. Namenjeno preverjanju kakovosti izluščenih značilk in označevanja.

Glavne funkcije:

- Nalaganje videa v pomnilnik
- Prikaz ključnih točk (MediaPipe)
- Prikaz trenutne faze iz anotacij (če so na voljo)
- Nadzor predvajanja: pavza, premikanje po posameznih kadrih

Nadzor:

- Preslednica: pavza / predvajanje
- a: korak nazaj (v pavzi)
- d: korak naprej (v pavzi)
- Esc: izhod

train_combined_dataset.py

Izuri model za klasifikacijo faz na podlagi koordinat ključnih točk. Uporablja se algoritem RandomForestClassifier iz knjižnice sklearn. Naučen model se shrani v datoteko za nadaljnjo uporabo.