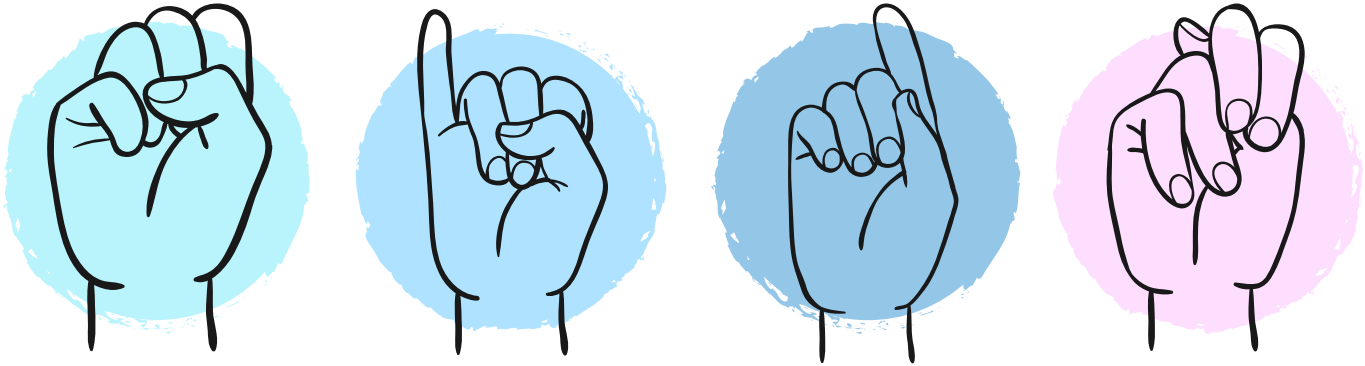
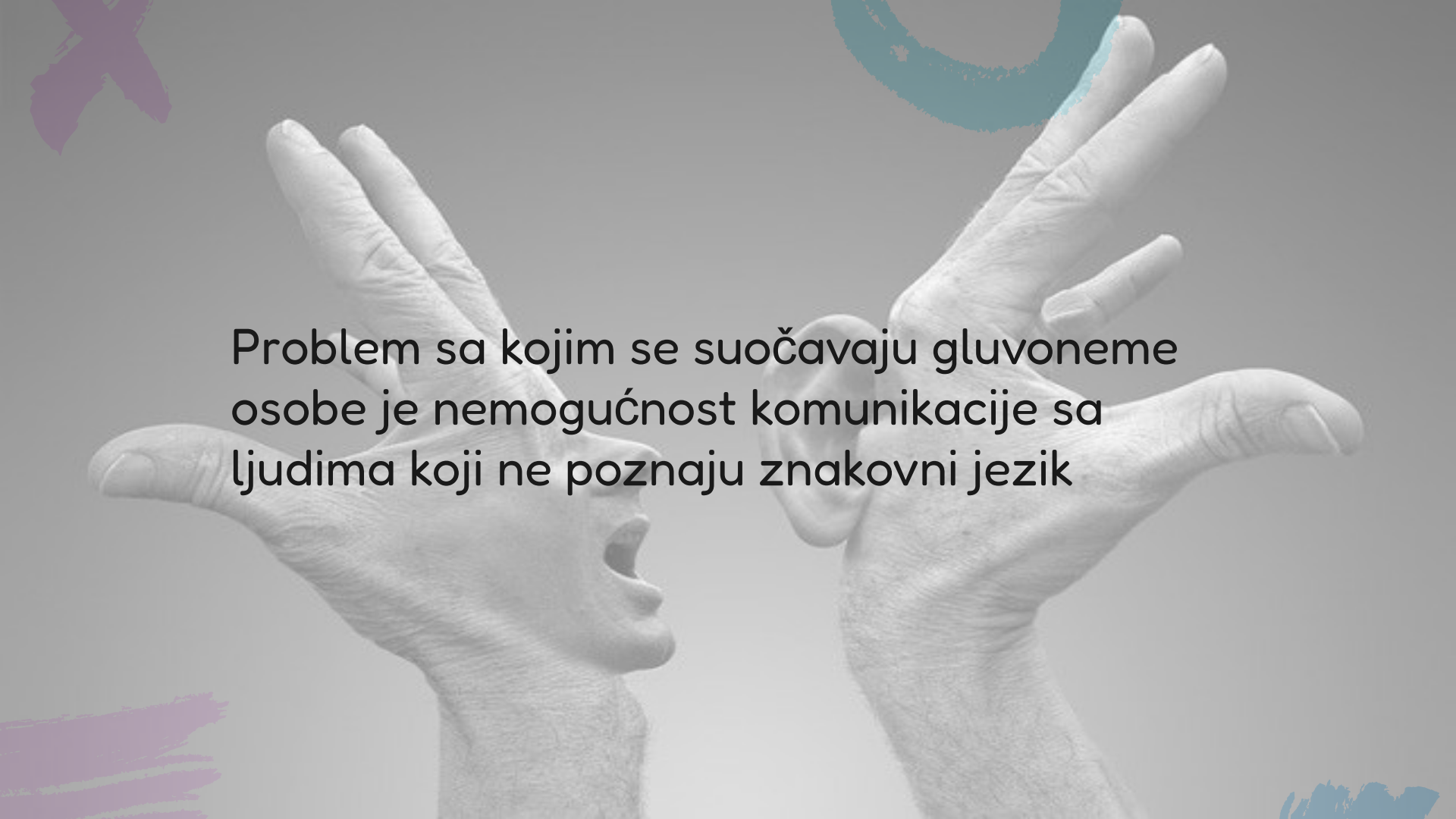
Sign language

Aplikacija za prevođenje znakovnog jezika u tekst ili govor u realnom vremenu

Jasmina Turku 1908





Problem sa kojim se suočavaju gluvoneme osobe je nemogućnost komunikacije sa ljudima koji ne poznaju znakovni jezik

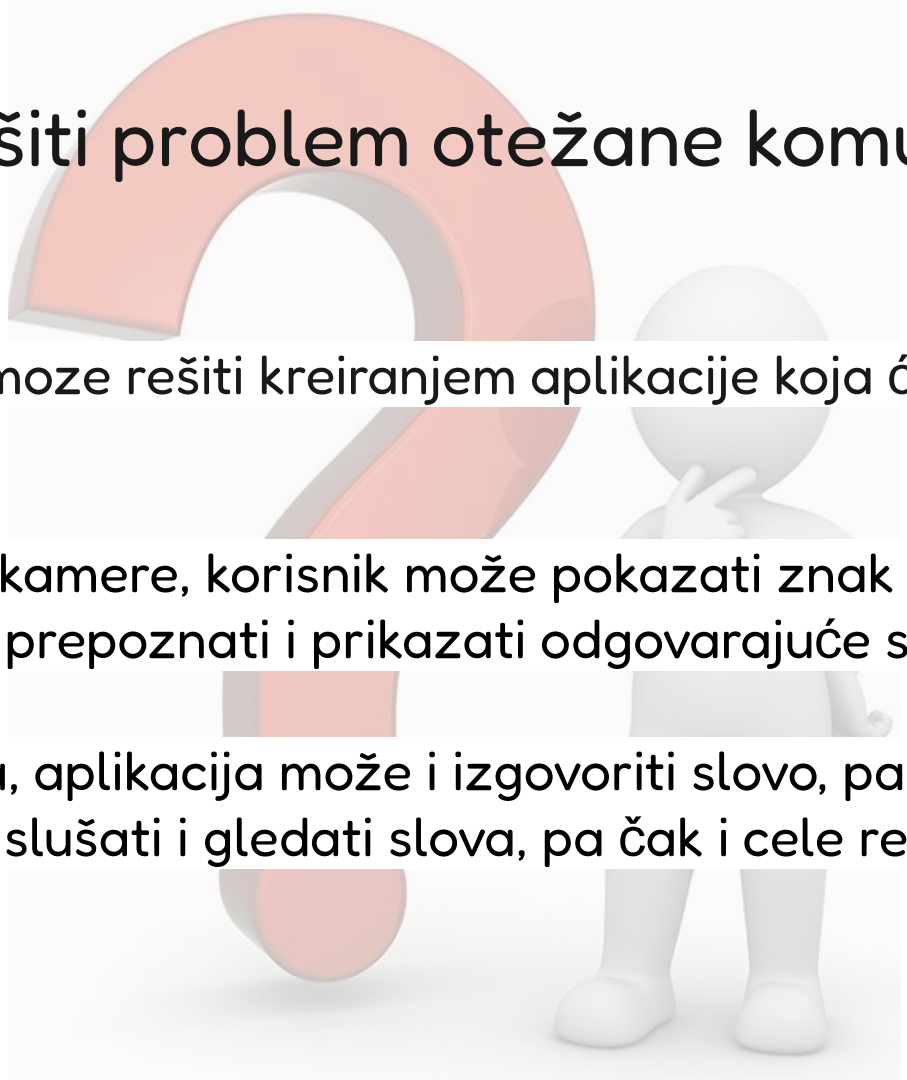
Otezana komunikacija

U Srbiji, prema podacima Saveza gluvih i nagluvih Srbije, ima oko 70.000 osoba kojima je prvi jezik - srpski znakovni jezik

U Srbiji ima oko 30 tumača znakovnog jezika za 70.000 ljudi

Nedostatak prevodilaca značajno otežava svakodnevni život osoba koje koriste znakovni jezik

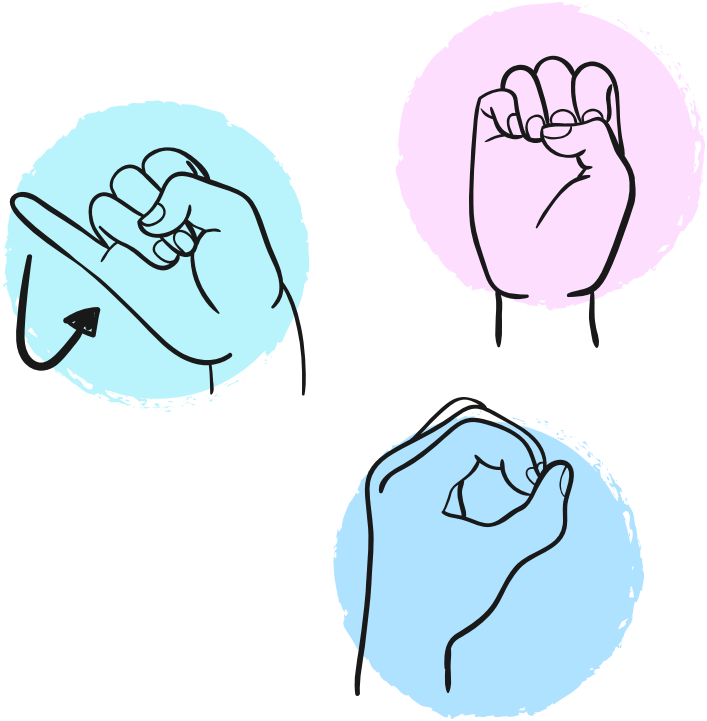
Kako rešiti problem otežane komunikacije?



Problem se moze rešiti kreiranjem aplikacije koja će biti dostupna svima

Korišćenjem kamere, korisnik može pokazati znak rukom, a aplikacija će prepoznati i prikazati odgovarajuće slovo

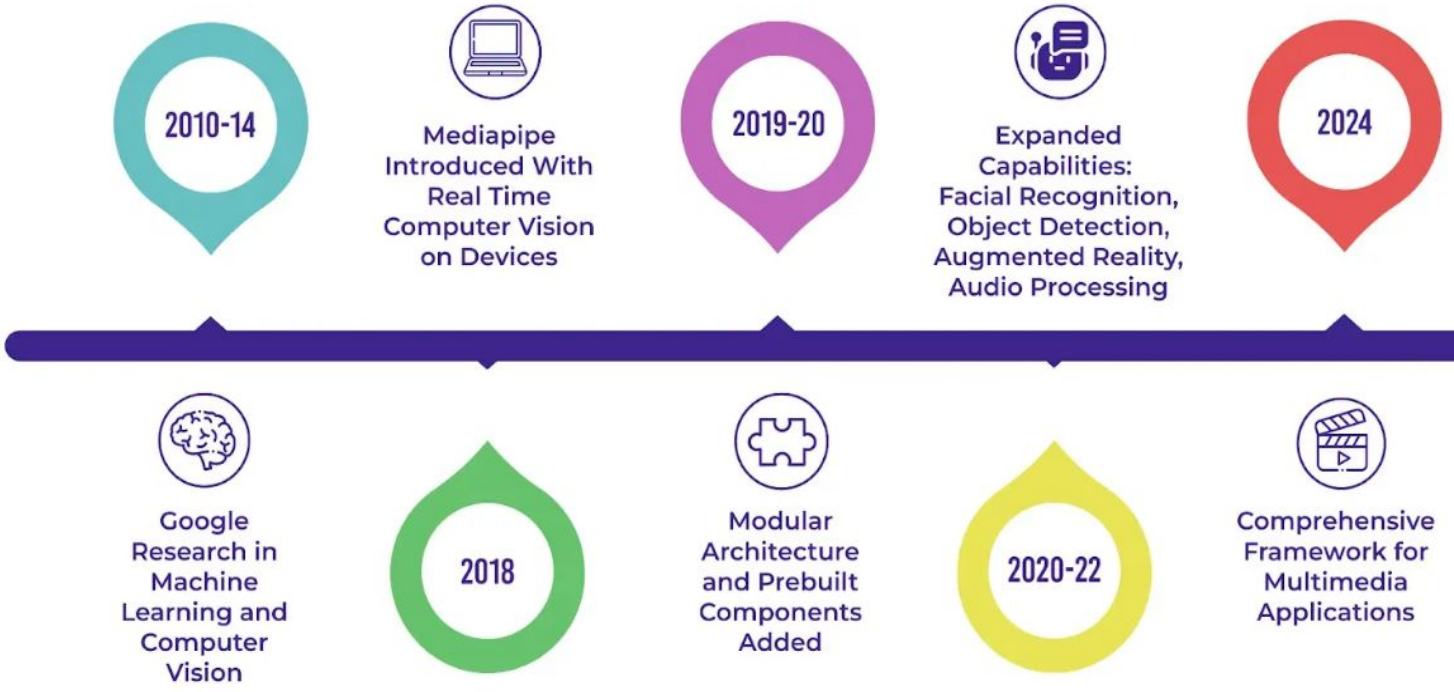
Osim prikaza, aplikacija može i izgovoriti slovo, pa korisnik može istovremeno slušati i gledati slova, pa čak i cele rečenice



Šta je MediaPipe?

Biblioteka koja omogućava praćenje pokreta prstiju i ruku u realnom vremenu, koristeći algoritme za prepoznavanje ključnih tačaka na ruci

Razvoj MediaPipe-a



Modularna ahitektura

MediaPipe je napravljena kao sklop manjih delova (modula), gde svaki modul obavlja jedan specifičan zadatak

Ti moduli se zovu kalkulatori (calculators) i mogu se povezivati u graf (pipeline)

Kalkulator - to je komponenta koja prima podatke, obrađuje ih i vraća rezultat

Pipeline (ili Dataflow Graph) je niz povezanih calculatora

Osnovne funkcionalnosti

MediaPipe dolazi sa različitim funkcijama:

Jedna od ključnih karakteristika je mogućnost korišćenja snage GPU-a za bržu obradu podataka

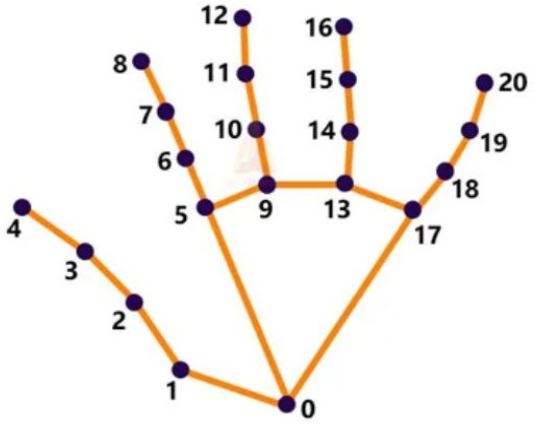
Zahvaljujući paralelnom procesiranju, MediaPipe može istovremeno obavljati više zadataka

MediaPipe moze da koristi OpenCV - moćna biblioteka za obradu slika i videa

MediaPipe Hand Landmarker

MediaPipe Hand Landmarker je paket koji u sebi sadrži dva modela:

- Model za detekciju dlana - pronalazi gde se ruka nalazi na slici
- Model za detekciju ključnij tačaka koji prepoznaje 21 ključnu tačku na ruci (vrhovi prstiju, zglobove i td)



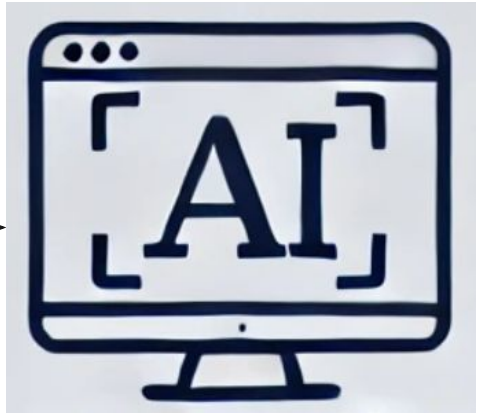
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Osobine	MediaPipe	OpenCV	TensorFlow
Nivo apstrakcije	Viši (gotovi moduli i pipeline-ovi)	Niži (niskonivska obrada slike)	Visoki i srednji (ML/DL framework)
Primarna svrha	Real-time računarski vid + gotovi ML modeli	Klasična obrada slike i videa	Treniranje i izvođenje ML/DL modela
Podrška za ML modele	Ograničena, koristi samo podržane modele	Ograničena (nije primarno za ML)	Potpuna – trening i izvođenje sopstvenih modela
Real-time performanse	Izuzetno dobre (GPU podrška, optimizacija)	Dobre, ali bez GPU podrške	Dobre
Gotovi alati/moduli	Da (ruke, lice, poze, govor...)	Ne – sve se pravi ručno	Ne – mora se trenirati ili učitati model
Fleksibilnost	Srednja – koristi se kroz grafe i kalkulatore	Visoka – puna kontrola nad svakim korakom	Veoma visoka – moguće praviti bilo koje modele

Pretvaranje znakova u tekst/govor

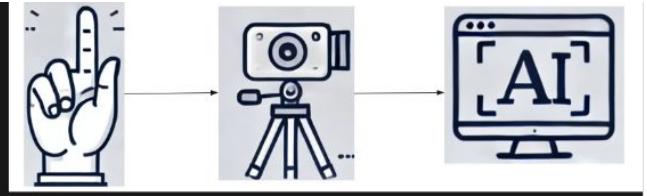
Ulazni podatak:
Prikazati znak

Kamera
detektuje znak



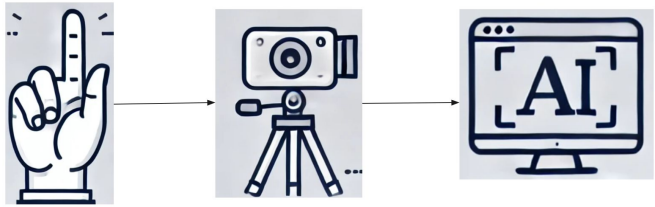
Izlazni podatak:
Na ekranu se prikazuje slovo

Struktura projekta

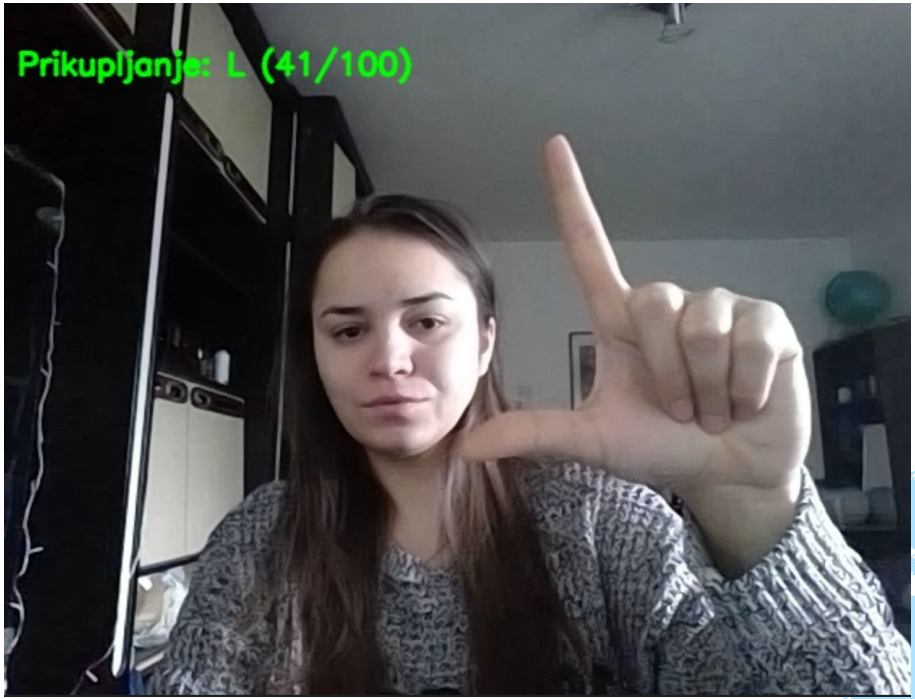


- data
- utils
  - __pycache__
  - create_features.py
  - draw_hand_landmarks.py
  - draw_predicted_character.py
  - extract_hand_landmark_coordinates.py
  - text_to_speech.py
- .gitignore
- 1_collect_imgs.py
- 2_create_dataset.py
- 3_train_classifier.py
- 4_inference_classifier.py
- data.pickle
- model.p

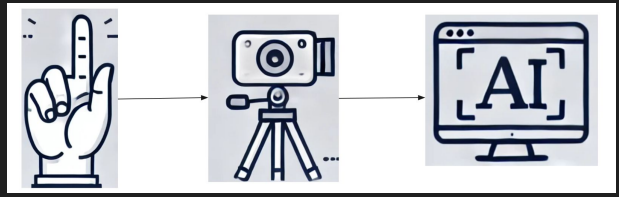
Prikupljanje slika



Ulazni podaci:

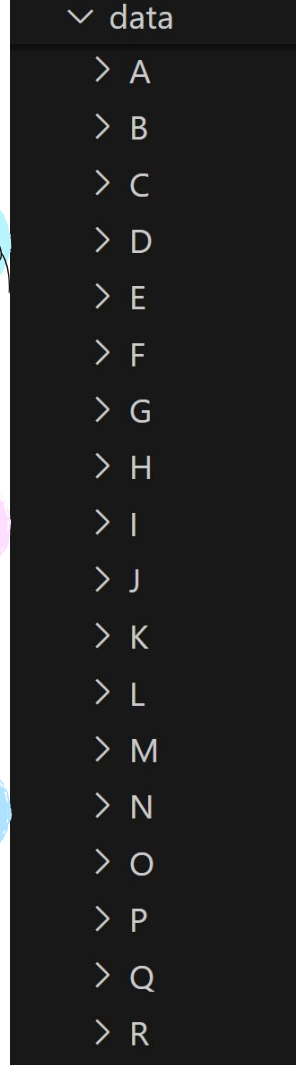


Prikupljanje slika

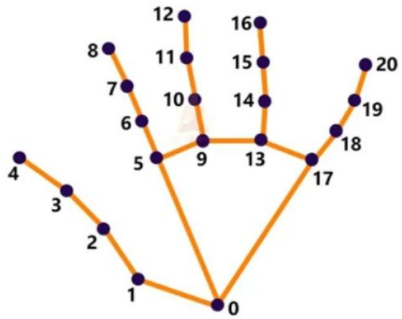


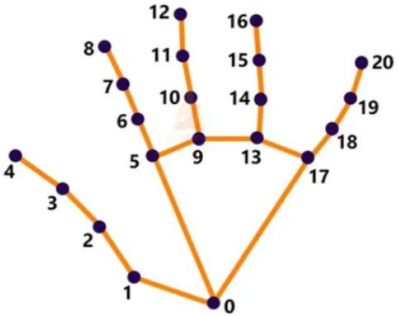

```
cap = cv2.VideoCapture(0)
counter = 0
while counter < dataset_size:
    ret, frame = cap.read()
    if not ret:
        continue

    letter = cv2.waitKey(1) & 0xFF
    letter_dir = os.path.join(DATA_DIR, letter)
    if not os.path.exists(letter_dir):
        os.makedirs(letter_dir)

    cv2.putText(frame, f'Prikupljanje: {letter} ({counter + 1}/{dataset_size})',
                (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2, cv2.LINE_AA)
    cv2.imshow('frame', frame)

    time.sleep(0.1)
    cv2.imwrite(os.path.join(letter_dir, f'{counter}.jpg'), frame)
    counter += 1
```

```python
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        features, x_, y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                x_, y_ = extract_hand_landmark_coordinates(hand_landmarks)
                features = create_features(hand_landmarks, x_, y_)

            data.append(features)
            labels.append(dir_)

f = open('data.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```



```python
def extract_hand_landmark_coordinates(hand_landmarks):
    x_ = []
    y_ = []
    for i in range(len(hand_landmarks.landmark)):
        x = hand_landmarks.landmark[i].x
        y = hand_landmarks.landmark[i].y
        x_.append(x)
        y_.append(y)
    return x_, y_
```






```python
max_length = 84
def create_features(hand_landmarks, x_, y_):
    features = []
    for i in range(len(hand_landmarks.landmark)):
        x = hand_landmarks.landmark[i].x
        y = hand_landmarks.landmark[i].y
        features.append(x - min(x_))
        features.append(y - min(y_))

    if len(features) < max_length:
        features.extend([0] * (max_length - len(features)))

    return features
```

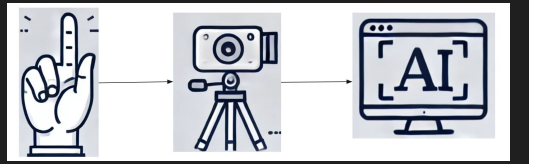
Treniranje modela

```
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import os

DATA_DIR = './data'
labels = sorted(os.listdir(DATA_DIR))
data_dict = pickle.load(open('./data.pickle', 'rb'))

label_mapping = {label: idx for idx, label in enumerate(labels)}
reverse_label_mapping = {idx: label for label, idx in label_mapping.items()}
filtered_data = []
filtered_labels = []

for data, label in zip(data_dict['data'], data_dict['labels']):
    if label in label_mapping:
        filtered_data.append(data)
        filtered_labels.append(label_mapping[label])
```

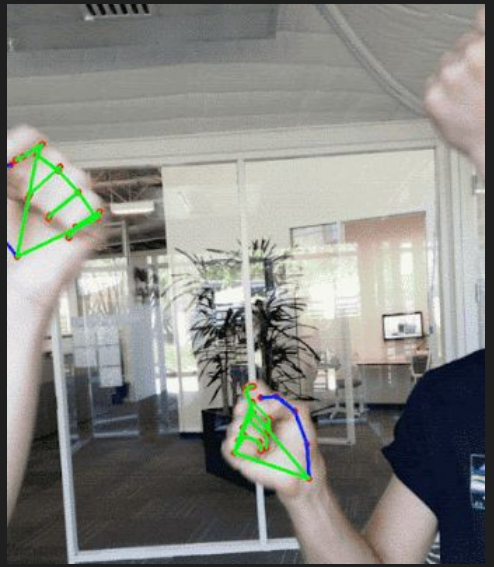
Treniranje modela

```
data = np.asarray(filtered_data)
labels = np.asarray(filtered_labels)

x_train, x_test, y_train, y_test = train_test_split(
    data, labels, test_size=0.2, shuffle=True, stratify=labels
)

model = RandomForestClassifier(random_state=84)
model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)
print(f'{score * 100:.2f}% of samples were classified correctly!')

with open('model.p', 'wb') as f:
    pickle.dump({'model': model, 'label_mapping': reverse_label_mapping}, f)
```

```
label_mapping = {label: idx for idx, label in enumerate(labels)}
reverse_label_mapping = {idx: label for label, idx in label_mapping.items()}

cap = cv2.VideoCapture(0)
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```



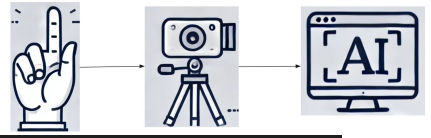
Prepoznavanje znakova



Prepoznavanje znakova

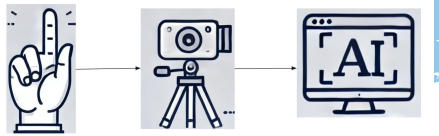
```
H, W, _ = frame.shape
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

results = hands.process(frame_rgb)
if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        draw_hand_landmarks(frame, hand_landmarks)

        x_, y_ = extract_hand_landmark_coordinates(hand_landmarks)
        features = create_features(hand_landmarks, x_, y_)

        prediction = model.predict([np.asarray(features)])
        predicted_character = reverse_label_mapping[int(prediction[0])]

        draw_predicted_character(frame, x_, y_, W, H, predicted_character)
        text_to_speech_threaded(predicted_character)

cv2.imshow('frame', frame)
```

Prikaz tačaka na ruci



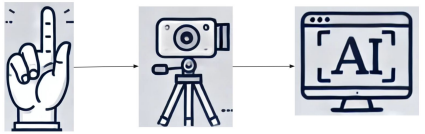

```python
import mediapipe as mp

def draw_hand_landmarks(frame, hand_landmarks):
    mp_drawing = mp.solutions.drawing_utils
    mp_drawing_styles = mp.solutions.drawing_styles
    mp_hands = mp.solutions.hands

    mp_drawing.draw_landmarks(
        frame,
        hand_landmarks,
        mp_hands.HAND_CONNECTIONS,
        mp_drawing_styles.get_default_hand_landmarks_style(),
        mp_drawing_styles.get_default_hand_connections_style())
```

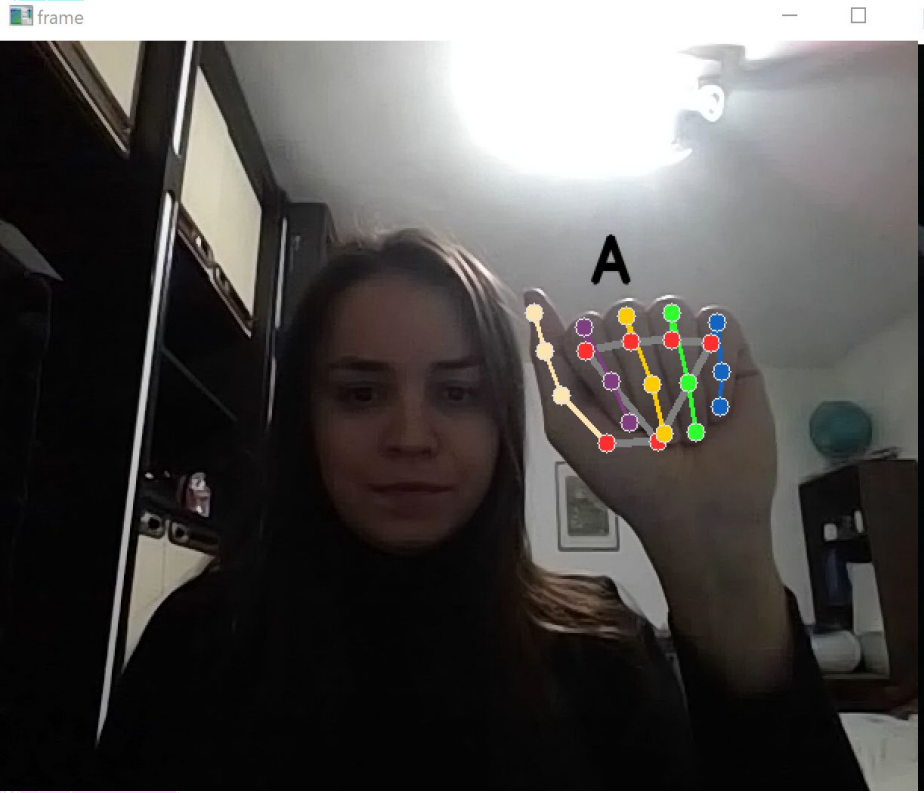
Prikaz prepoznatog karaktera

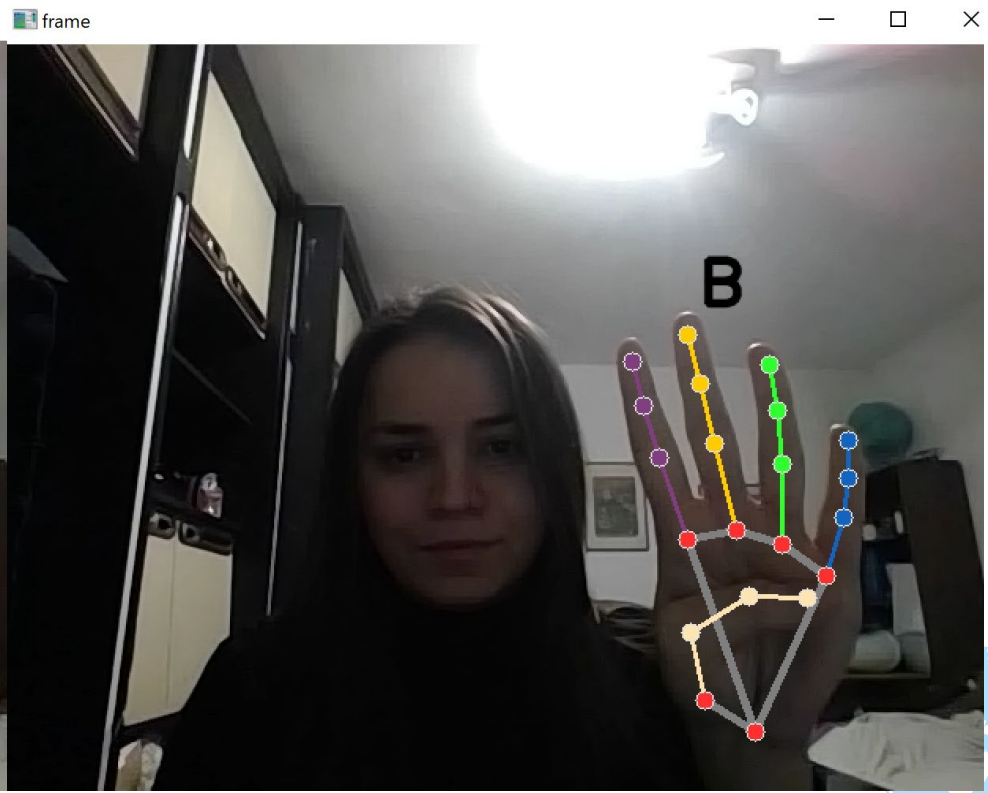



```python
import cv2

def draw_predicted_character(frame, xCords, yCords, W, H, predicted_character):
    x1 = int(min(xCords) * W) - 10
    y1 = int(min(yCords) * H) - 10
    x2 = int(max(xCords) * W) - 10

    center_x = (x1 + x2) // 2

    text_width, text_height = cv2.getTextSize(predicted_character, cv2.FONT_HERSHEY_SIMPLEX, 1.3,
    text_x = center_x - text_width // 2
    text_y = max(y1 - 10, text_height)

    cv2.putText(frame, predicted_character, (text_x, text_y),
                cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3, cv2.LINE_AA)
```

Kao izlaz dobijamo







Hvala na pažnji!