ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΙΕΣ

ΨΗΦΙΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΒΙΝΤΕΟ

ΑΣΚΗΣΗ 3

Ονοματεπώνυμο:	Βασιλική Στάμου
Αριθμός Μητρώου:	1059543
Τομέας:	Μεταπτυχιακό ΣΜΗΝ
Έτος φοίτησης:	1ο
Ακ. έτος διεξαγωγής μαθήματος:	2023-2024
Ημ/νία διεξαγωγής Φ.Α.:	03.04.2024

Περιεχόμενα

1 Υλοποιήστε μια συνάρτηση που λαμβάνει ένα βίντεο που αναπαριστά μία κίνηση και εξάγει motion energy image. Χρησιμοποιείστε τη συνάρτηση ώστε να υλοποιήσετε ένα ταξινομητή που θα κάνει ταξινόμηση σε μία από τις 4 κινήσεις με χρήση motion energy images.

Στην παρούσα εργασία, εστιάζουμε στην ταξινόμηση κινήσεων που καταγράφονται μέσω βίντεο. Η εργασία περιλαμβάνει την εξαγωγή Motion Energy Images (MEI) από βίντεο χειρονομιών και την ταξινόμηση αυτών των εικόνων χρησιμοποιώντας τον αλγόριθμο k-Nearest Neighbors (k-NN). Η βάση δεδομένων περιλαμβάνει τέσσερις κατηγορίες χειρονομιών: 'Clic', 'No', 'Rotate', και 'StopGraspOk'. Η εργασία απαιτεί την υλοποίηση ταξινομητή με χρήση των πρώτων 5 χειρονομιών από κάθε κατηγορία για training και των υπόλοιπων για testing.

Υλοποίηση:

1. **Εξαγωγή Motion Energy Image (MEI):** Δημιουργούμε μια συνάρτηση extractMEI, η οποία εξάγει το Motion Energy Image (MEI) από μια σειρά εικόνων βίντεο. Η συνάρτηση υπολογίζει τη διαφορά μεταξύ διαδοχικών εικόνων και δημιουργεί μια εικόνα που απεικονίζει την ενέργεια κίνησης.
2. **Δημιουργία Training και Testing Συνόλων:** Χρησιμοποιούμε τις πρώτες 5 εικόνες από κάθε κατηγορία για training και τις υπόλοιπες για testing. Δημιουργούμε πίνακες δεδομένων για training και testing με τις αντίστοιχες ετικέτες.
3. **Training και Testing με k-NN:** Εφαρμόζουμε τον αλγόριθμο k-Nearest Neighbors (k-NN) με αρχικό k = 3 και εκπαιδεύουμε το μοντέλο με τα training δεδομένα. Στη συνέχεια, αξιολογούμε την απόδοση του μοντέλου με τη χρήση των testing δεδομένων.
4. **Αξιολόγηση Μοντέλου:** Υπολογίζουμε το confusion matrix για k = 3 και δημιουργούμε heatmap για την οπτική απεικόνιση της απόδοσης του μοντέλου. Δοκιμάζουμε διαφορετικές τιμές του k από 1 έως 10 και εξετάζουμε την επίδραση στο accuracy.

Αποτελέσματα για k=3:

```
Overall Accuracy: 83.7838%
Per-Class Accuracy:
    1.0000
    1.0000
    0.7500
    0.6000
```

Figure 1. Συνολικό και ανά κλάση Accuracy για k=3.

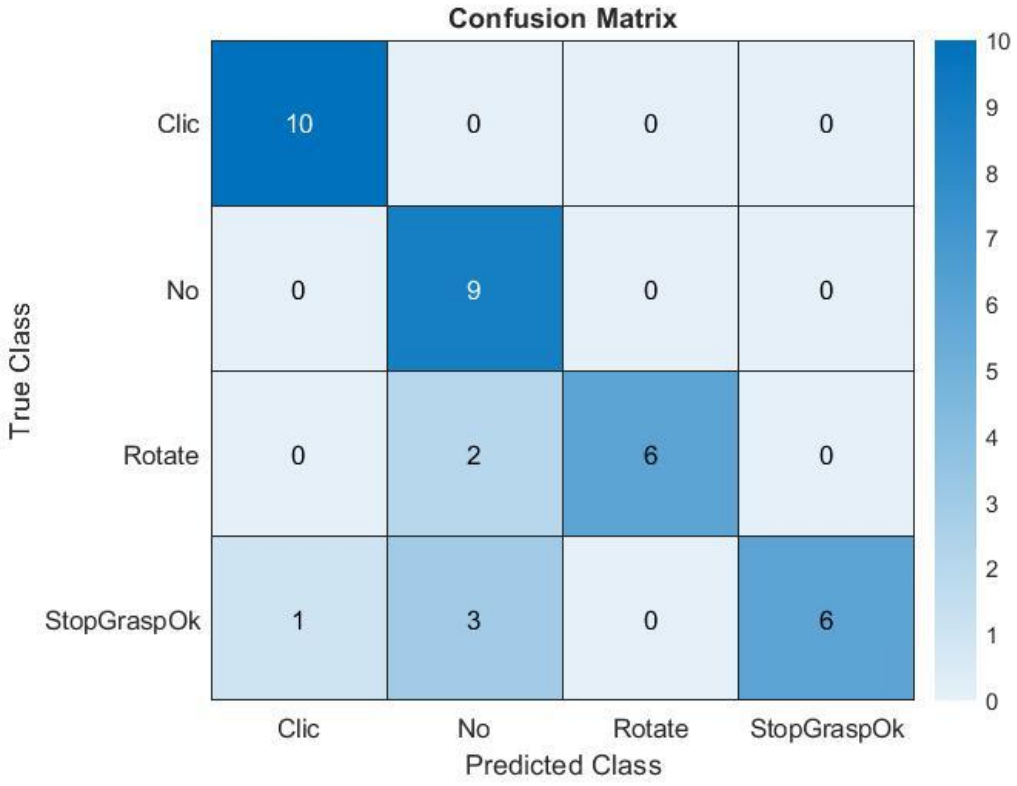


Figure 2. Heatmap Confusion Matrix για k=3.

Παρατηρήσεις:

- Ο αλγόριθμος k-NN με k = 3 έδειξε πολύ καλά αποτελέσματα για την κατηγορία 'Clic', όπου όλες οι testing εικόνες ταξινομήθηκαν σωστά.
- Για κάποιες κατηγορίες όπως η 'StopGraspOk', παρατηρήθηκε υψηλός αριθμός λανθασμένων ταξινομήσεων. Αυτό υποδηλώνει ότι το μοντέλο έχει δυσκολία στη διάκριση αυτής της κατηγορίας από άλλες.
- Ο ταξινομητής έχει ισχυρή απόδοση για τις κατηγορίες 'Clic' και 'No', με μηδενικές λανθασμένες ταξινομήσεις. Οι κατηγορίες 'Rotate' και 'StopGraspOk' έχουν περισσότερα λανθασμένα αποτελέσματα, υποδεικνύοντας περιοχές που χρήζουν βελτίωσης.

Αποτελέσματα για διάφορες τιμές του k:

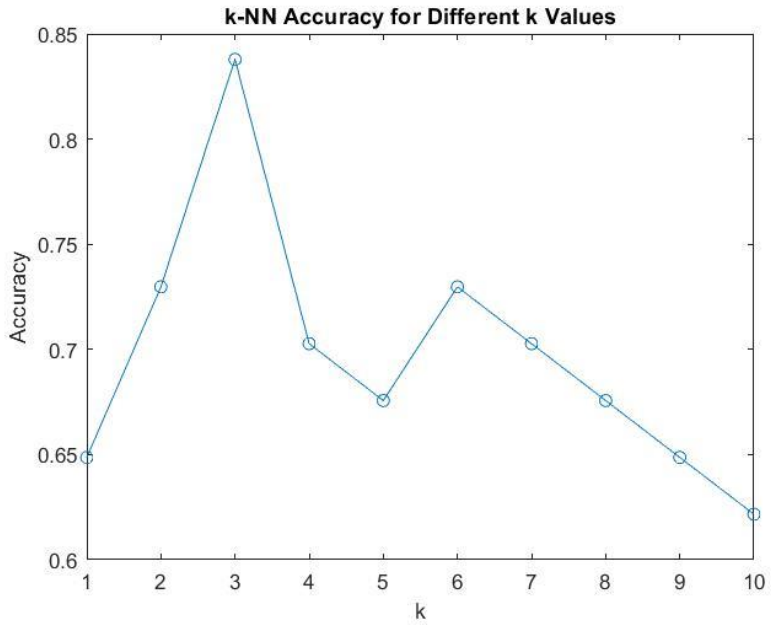
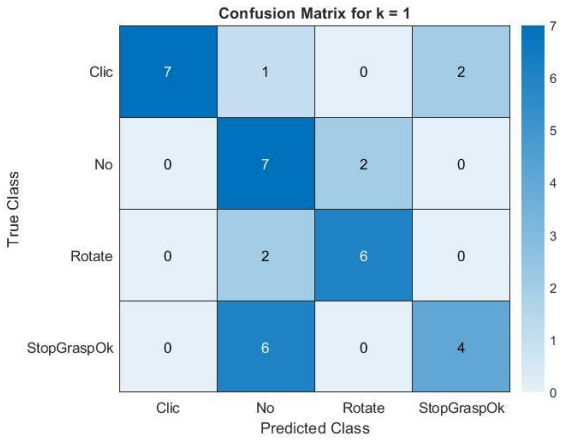
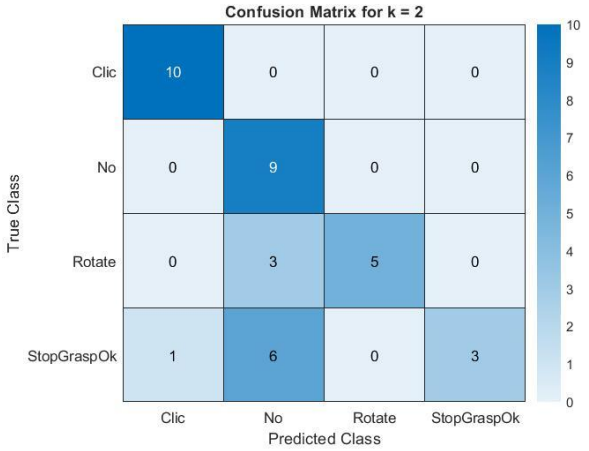
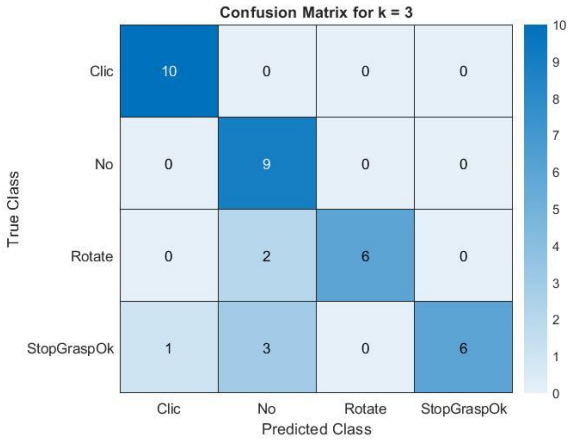


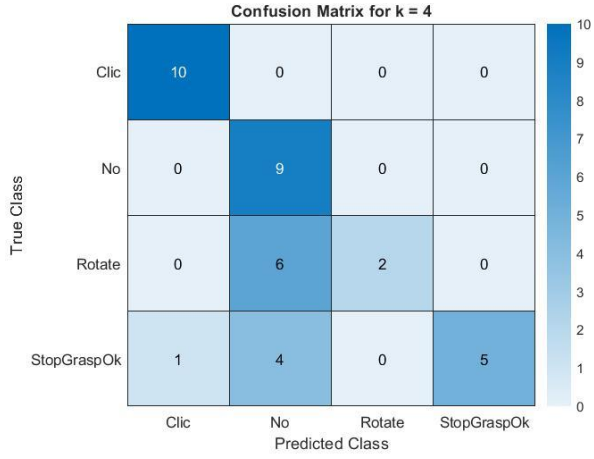
Figure 3. Accuracy για διάφορες τιμές του k.

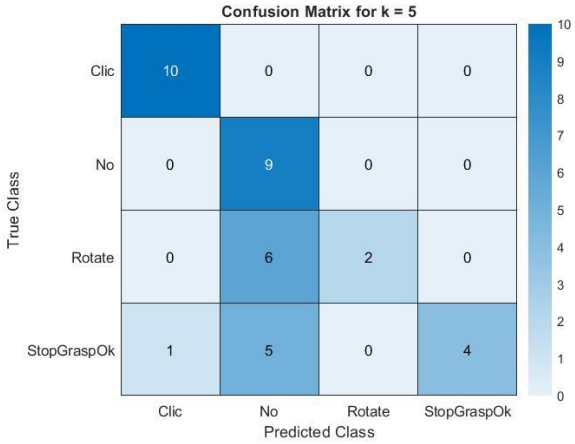
Οι αλλαγές στην τιμή του k επηρεάζουν την απόδοση του μοντέλου. Σε γενικές γραμμές, η ακρίβεια αυξάνεται με μικρές τιμές του k, αλλά η βέλτιστη τιμή διαφέρει ανάλογα με τα δεδομένα και τις κατηγορίες, για την περίπτωση μας η βέλτιστη τιμή είναι για k=3.

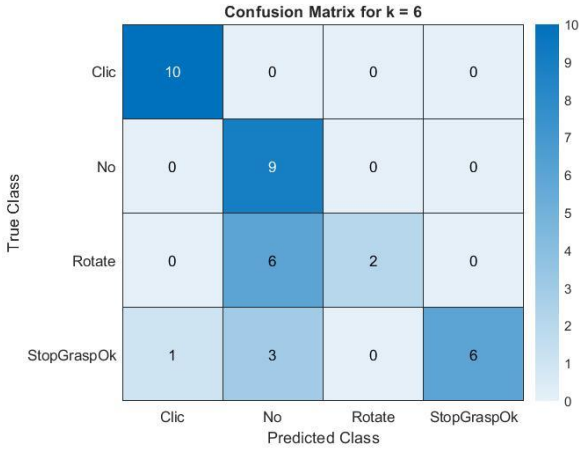

Confusion Matrix for k = 1
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
7 1 0 2
0 7 2 0
0 2 6 0
0 6 0 4


Confusion Matrix for k = 2
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
10 0 0 0
0 9 0 0
0 3 5 0
1 6 0 3


Confusion Matrix for k = 3
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
10 0 0 0
0 9 0 0
0 2 6 0
1 3 0 6


Confusion Matrix for k = 4
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
10 0 0 0
0 9 0 0
0 6 2 0
1 4 0 5


Confusion Matrix for k = 5
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
10 0 0 0
0 9 0 0
0 6 2 0
1 5 0 4


Confusion Matrix for k = 6
True Class
Predicted Class
Clic
No
Rotate
StopGraspOk
10 0 0 0
0 9 0 0
0 6 2 0
1 3 0 6

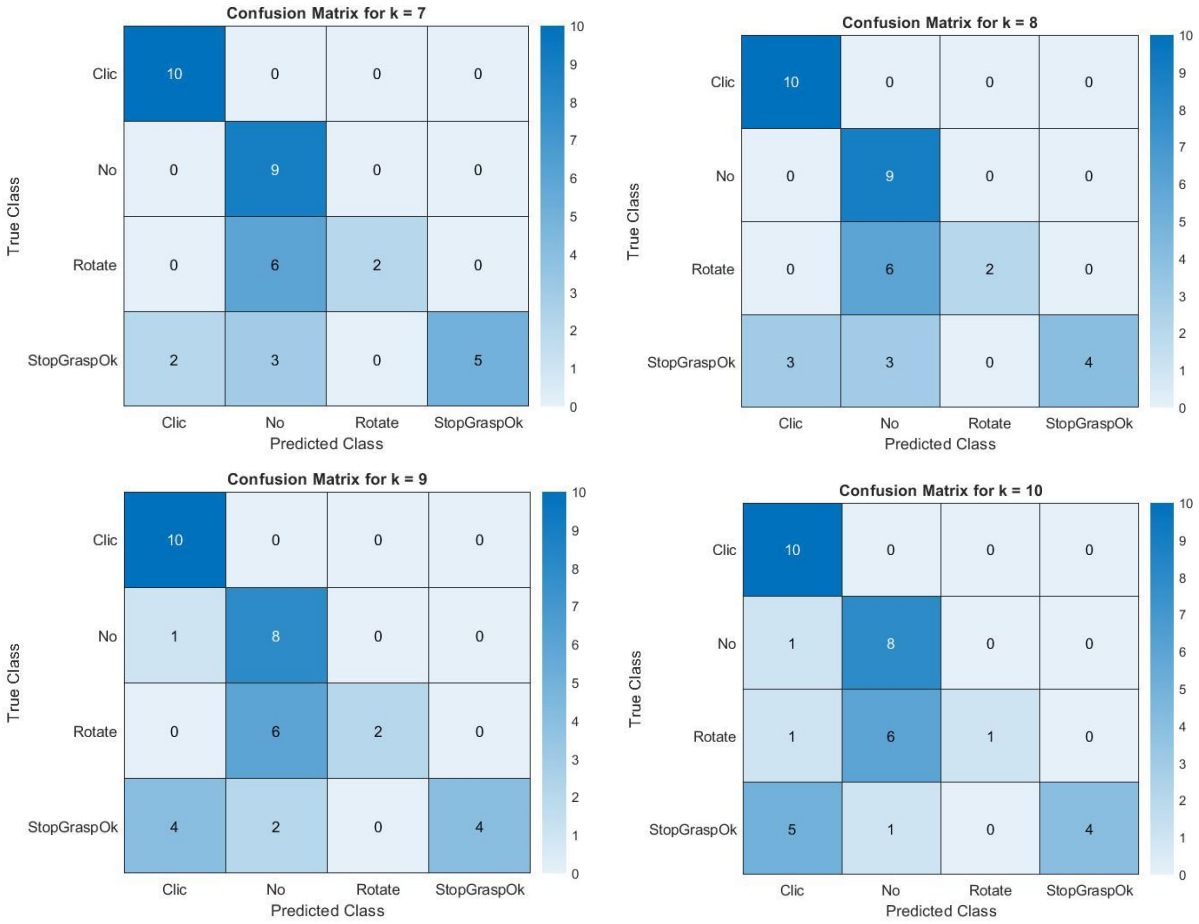


Figure 4. Heatmap Confusion Matrix για k=1-10.

2 Υλοποιήστε ένα ταξινομητή με χρήση της τεχνικής dynamic time warping.

Σκοπός του ερωτήματος είναι η ταξινόμηση χειρονομιών χρησιμοποιώντας την τεχνική Dynamic Time Warping (DTW) και τον αλγόριθμο k-Nearest Neighbors (k-NN). Οι χειρονομίες κατηγοριοποιούνται σε τέσσερις κατηγορίες: "Clic", "No", "Rotate", και "StopGraspOk". Κάθε κατηγορία περιλαμβάνει μια σειρά από εικόνες που απεικονίζουν την κίνηση του χεριού.

Για την εκπαίδευση του συστήματος χρησιμοποιούνται οι πρώτες 5 κινήσεις από κάθε κατηγορία, ενώ για την αξιολόγηση χρησιμοποιούνται οι υπόλοιπες κινήσεις. Η μέθοδος περιλαμβάνει την εξαγωγή χαρακτηριστικών από τις εικόνες που αφορούν είτε τη θέση του χεριού (i,j) είτε τη θέση και το σχήμα (διαστάσεις του ορθογωνίου που περιβάλλει το χέρι).

Υλοποίηση:

1. **Απομόνωση του Χεριού από το Υπόβαθρο**: Αρχικά, κάθε εικόνα της χειρονομίας επεξεργάζεται για να διαχωριστεί το χέρι από το υπόβαθρο. Αυτό επιτυγχάνεται μέσω της αφαίρεσης του οπτικού υποβάθρου, το οποίο έχει ενιαίο χρώμα. Η διαδικασία περιλαμβάνει:
    - Αναδιάταξη της εικόνας σε έναν πίνακα 2D όπου κάθε σειρά αντιστοιχεί σε ένα εικονοστοιχείο και οι στήλες στα χρώματα RGB.
    - Εκτίμηση του χρώματος του υποβάθρου ως το πιο κοινό χρώμα στην εικόνα.
    - Υπολογισμός της Ευκλείδιας απόστασης κάθε εικονοστοιχείου από το χρώμα του υποβάθρου.
    - Δημιουργία μιας δυαδικής μάσκας που υποδεικνύει τα εικονοστοιχεία που ανήκουν στο χέρι.
2. **Εξαγωγή Χαρακτηριστικών:** Από τη δυαδική μάσκα που δημιουργήθηκε στο προηγούμενο βήμα, εξάγονται τα χαρακτηριστικά του χεριού. Υπάρχουν δύο επιλογές για την αναπαράσταση του χεριού:
    - Θέση (i,j): Το χέρι αναπαρίσταται από τη θέση του κέντρου βάρους του στη δυαδική εικόνα.
    - Θέση και Σχήμα (διαστάσεις ορθογωνίου): Εκτός από τη θέση, εξάγονται και οι διαστάσεις του ορθογωνίου που περιβάλλει το χέρι (πλάτος και ύψος). Αυτά τα χαρακτηριστικά κανονικοποιούνται για να αποφευχθούν διαφορές στην κλίμακα που μπορεί να επηρεάσουν τον υπολογισμό της απόστασης.
3. **Υπολογισμός Απόστασης DTW:** Για την ταξινόμηση των χειρονομιών, χρησιμοποιείται η μέθοδος Dynamic Time Warping (DTW) για τον υπολογισμό της απόστασης μεταξύ των ακολουθιών χαρακτηριστικών των εικόνων. Η DTW επιτρέπει την εύρεση της βέλτιστης αντιστοίχισης μεταξύ δύο ακολουθιών χρόνου, ακόμα και αν αυτές έχουν διαφορετικά μήκη ή μη ομοιόμορφες ταχύτητες.
4. **Ταξινόμηση με k-NN:** Με βάση τις αποστάσεις DTW, εφαρμόζεται ο αλγόριθμος k-Nearest Neighbors (k-NN) για την πρόβλεψη της κατηγορίας της χειρονομίας. Για κάθε testing δείγμα:
    - Υπολογίζονται οι αποστάσεις DTW μεταξύ του testing δείγματος και όλων των training δειγμάτων.
    - Τα k κοντινότερα γειτονικά δείγματα επιλέγονται με βάση αυτές τις αποστάσεις.
    - Η κατηγορία του δείγματος προβλέπεται με πλειοψηφική ψήφο των κατηγοριών των k γειτόνων.
5. **Αξιολόγηση και Σύγκριση Αποτελεσμάτων:** Η απόδοση του ταξινομητή αξιολογείται για διάφορες τιμές του k (1 έως 10). Η ακρίβεια υπολογίζεται για κάθε τιμή του k, και δημιουργούνται confusion matrices που δείχνουν την απόδοση της ταξινόμησης για κάθε κατηγορία χειρονομίας. Αυτό επιτρέπει την αναγνώριση των τιμών του k που παρέχουν τα καλύτερα αποτελέσματα.

Ερώτημα (α) - αποτελέσματα για k=3:

```
Overall Accuracy: 45.9459%
Per-Class Accuracy:
    0.7000
    0.6667
    0.1250
    0.3000
```

Figure 5. Συνολικό και ανά κλάση Accuracy για k=3, (α).

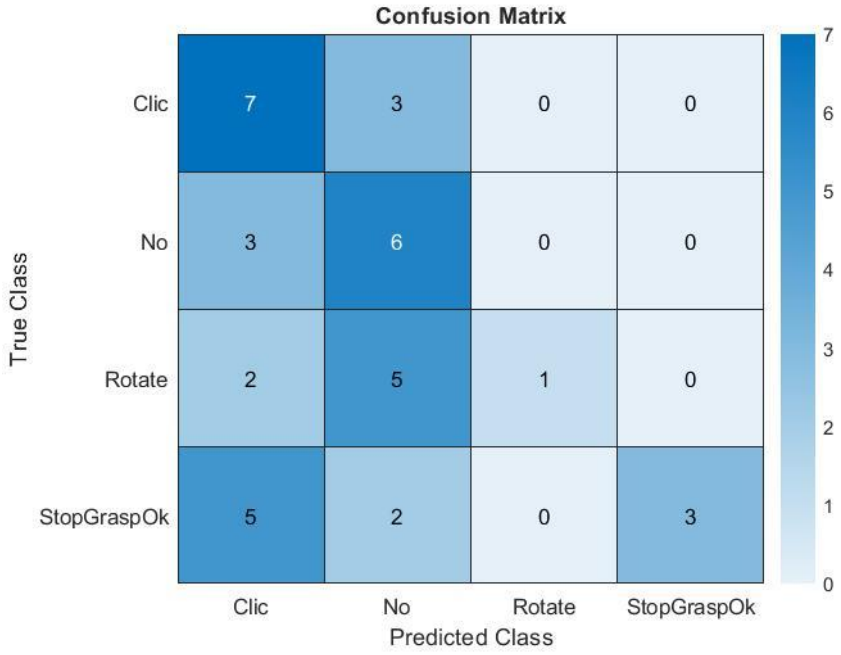


Figure 6. Heatmap Confusion Matrix για k=3, (α).

Παρατηρήσεις:

- Ο αλγόριθμος k-NN με k = 3 έδειξε μέτρια αποτελέσματα για τις κατηγορίες 'Clic' και 'No' .
- Για τις κατηγορίες 'Rotate' και 'StopGraspOk', παρατηρήθηκε υψηλός αριθμός λανθασμένων ταξινομήσεων. Αυτό υποδηλώνει ότι το μοντέλο έχει δυσκολία στη διάκριση αυτών των κατηγοριών από άλλες.

Ερώτημα (α) - αποτελέσματα για διάφορες τιμές του k:

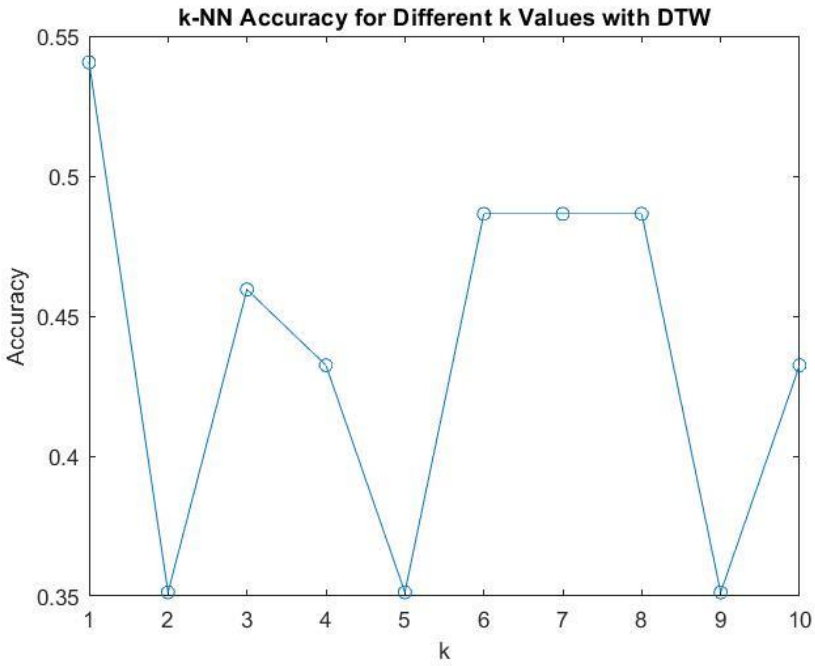


Figure 7. Accuracy για διάφορες τιμές του k, (α).

Οι αλλαγές στην τιμή του k επηρεάζουν την απόδοση του μοντέλου. Σε γενικές γραμμές η βέλτιστη τιμή διαφέρει ανάλογα με τα δεδομένα και τις κατηγορίες, για την περίπτωση μας η βέλτιστη τιμή είναι για k=1.

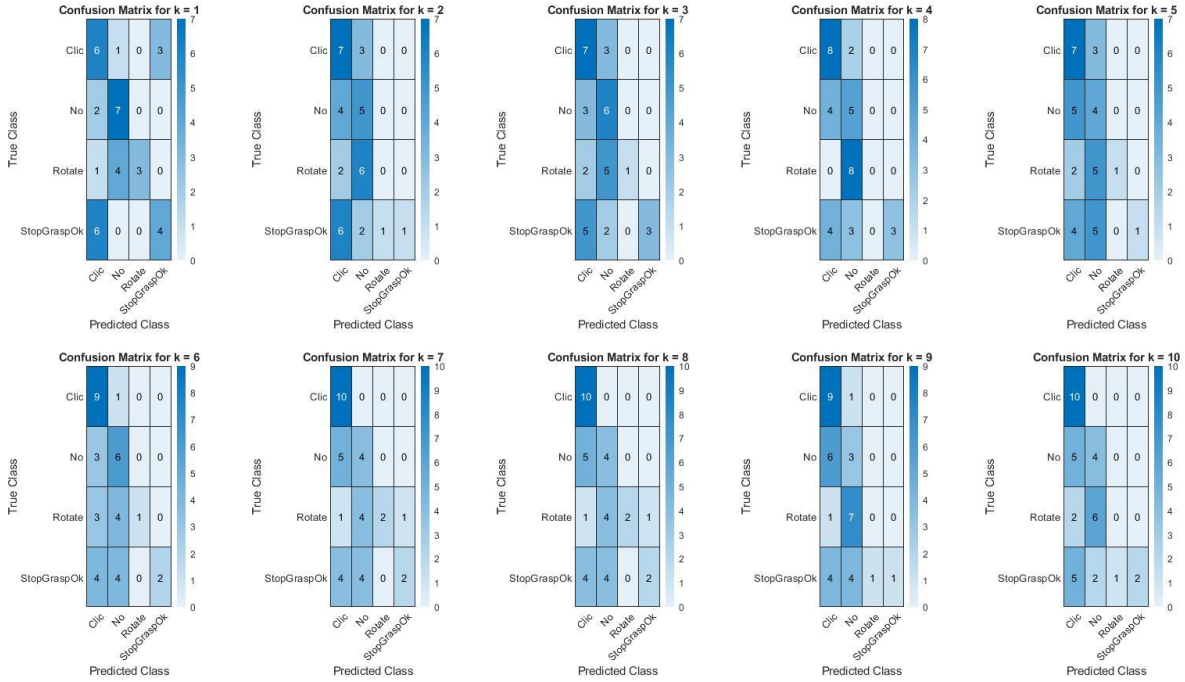


Figure 8. Heatmap Confusion Matrix για k=1-10, (α).

Ερώτημα (β) - αποτελέσματα για k=3:

```
Overall Accuracy: 83.7838%
Per-Class Accuracy:
    0.9000
    1.0000
    0.8750
    0.6000
```

Figure 9. Συνολικό και ανά κλάση Accuracy για k=3, (β).

Έχουμε ακριβώς το ίδιο accuracy με τον ταξινομητή που υλοποιήθηκε με moving energy image, το per-class accuracy ωστόσο είναι διαφορετικό για τους δύο αυτούς ταξινομητές με την κατηγοριά 'StopGraspOk' να εξακολουθεί να έχει 60% accuracy.

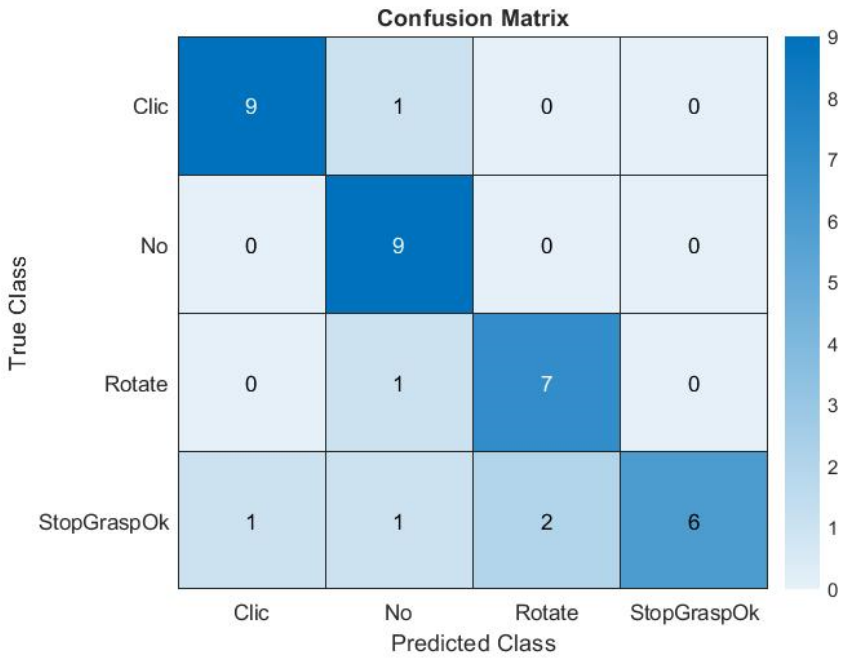


Figure 10. Heatmap Confusion Matrix για k=3, (β).

Παρατηρήσεις:

- Ο ταξινομητής έχει ισχυρή απόδοση για τις κατηγορίες 'Clic' και 'No', με μια και καμία λανθασμένη ταξινόμηση αντίστοιχα. Οι κατηγορίες 'Rotate' και 'StopGraspOk' έχουν περισσότερα λανθασμένα αποτελέσματα, υποδεικνύοντας περιοχές που χρήζουν βελτίωσης.

Ερώτημα (β) - αποτελέσματα για διάφορες τιμές του k:

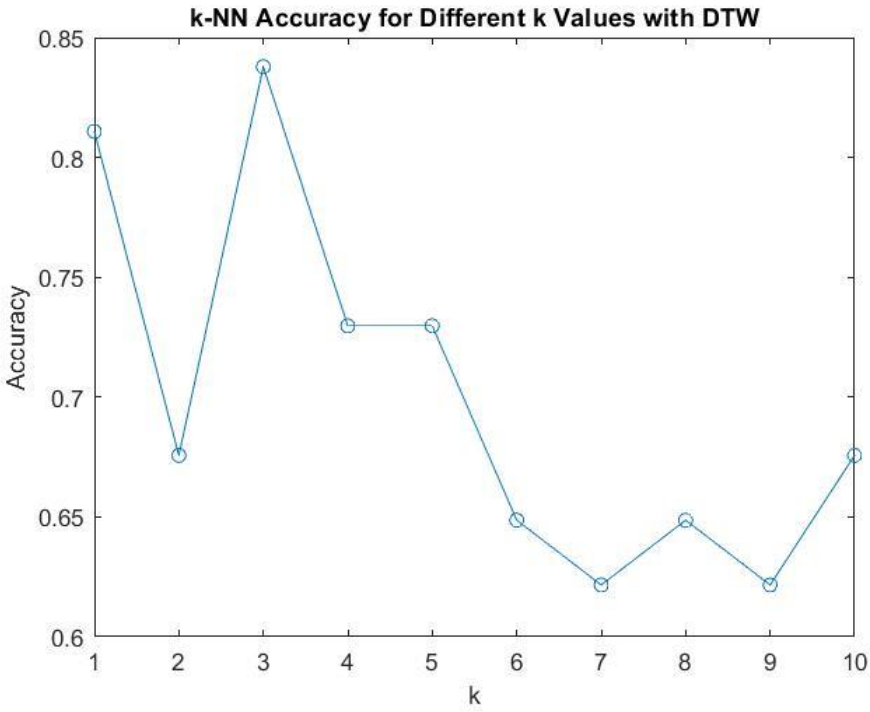


Figure 11. Accuracy για διάφορες τιμές του k, (β).

Οι αλλαγές στην τιμή του k επηρεάζουν την απόδοση του μοντέλου. Σε γενικές γραμμές η βέλτιστη τιμή διαφέρει ανάλογα με τα δεδομένα και τις κατηγορίες, για την περίπτωση μας η βέλτιστη τιμή είναι για k=3.

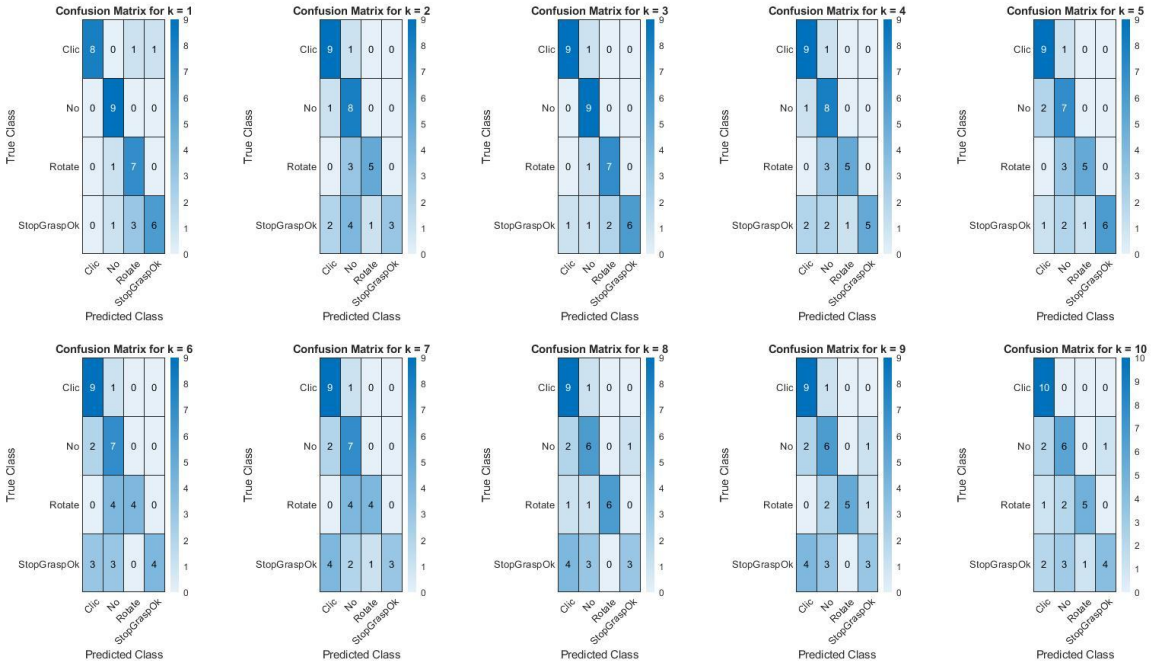


Figure 12. Heatmap Confusion Matrix για k=1-10, (β).

Εκτίμηση για τις Προσεγγίσεις (α) και (β) στη Χειρονομία:

Στο πλαίσιο της αναγνώρισης χειρονομιών, εξετάζονται δύο διαφορετικές αναπαραστάσεις του χεριού για να αξιολογηθεί η αποτελεσματικότητά τους στη διαδικασία ταξινόμησης.

(α) Αναπαράσταση με θέση (i, j):

Στην προσέγγιση αυτή, το χέρι αναπαρίσταται μόνο από τη θέση του, δηλαδή από τις συντεταγμένες (i, j) του κέντρου μάζας του. Αυτή η μέθοδος επικεντρώνεται αποκλειστικά στη χωρική θέση του χεριού χωρίς να λαμβάνει υπόψη το σχήμα ή το μέγεθος του χεριού. Το πλεονέκτημα αυτής της μεθόδου είναι η απλότητα και η ταχύτητα υπολογισμού, αφού η θέση (i, j) είναι ένα απλό και εύκολα υπολογίσιμο χαρακτηριστικό. Ωστόσο, είναι λιγότερο ακριβής σε περιπτώσεις όπου το σχήμα ή το μέγεθος του χεριού διαδραματίζουν σημαντικό ρόλο στην αναγνώριση της χειρονομίας για αυτό και έδωσε πολύ φτωχά αποτελέσματα.

(β) Αναπαράσταση με θέση (i, j) και σχήμα:

Στην προσέγγιση αυτή, το χέρι αναπαρίσταται τόσο από τη θέση (i, j) όσο και από το σχήμα του, όπως π.χ. από τις διαστάσεις του ορθογωνίου που το περιβάλλει. Αυτή η μέθοδος παρέχει επιπλέον πληροφορίες σχετικά με το μέγεθος και τη μορφολογία του χεριού, κάτι που είναι κρίσιμο για τη διάκριση χειρονομιών που έχουν παρόμοιες θέσεις αλλά διαφορετικά σχήματα.

Η χρήση επιπρόσθετων χαρακτηριστικών, όπως οι διαστάσεις, μπορεί να βελτιώσει την ακρίβεια της ταξινόμησης, ιδίως σε περιπτώσεις όπως η δική μας όπου το σχήμα του χεριού είναι διακριτό μεταξύ των χειρονομιών.

Εκτίμηση Αποτελεσματικότητας:

Η προσέγγιση (β), που συνδυάζει τη θέση με το σχήμα του χεριού, δίνει καλύτερα αποτελέσματα, ιδιαίτερα σε σύνθετες χειρονομίες όπου η διαφορά στη μορφή και το μέγεθος είναι σημαντική. Αντίθετα, η προσέγγιση (α) μπορεί να λειτουργήσει καλά σε πιο απλές χειρονομίες όπου η θέση είναι το κυρίαρχο χαρακτηριστικό. Ωστόσο, η προσέγγιση (β) απαιτεί περισσότερους υπολογιστικούς πόρους και να είναι πιο ευαίσθητη σε θόρυβο ή παραμορφώσεις στην εικόνα.

Γενικές παρατηρήσεις

Στην αναγνώριση χειρονομιών, η χρήση της τεχνικής **Motion Energy Image (MEI)** και η τεχνική **Dynamic Time Warping (DTW)** αποτελούν δύο διαφορετικές προσεγγίσεις για την ταξινόμηση των κινήσεων. Η **ταξινόμηση με MEI** βασίζεται στην αποτύπωση της κίνησης ως μια εικόνα που συνοψίζει την ενεργειακή κατανομή της κίνησης στο χρόνο. Αυτή η μέθοδος μπορεί να είναι αποτελεσματική σε περιπτώσεις όπου η γενική κατεύθυνση και η ταχύτητα της κίνησης είναι σημαντικοί παράγοντες. Ωστόσο, η **ταξινόμηση με DTW** επικεντρώνεται στη σύγκριση ακολουθιών κίνησης με βάση τη χρονική παραμόρφωση, επιτρέποντας την αντιστοίχιση ακολουθιών διαφορετικού μήκους και τη λήψη υπόψη περισσότερων χαρακτηριστικών, όπως η θέση και το σχήμα του χεριού (στην περίπτωση της προσέγγισης (β)).

Παρόλο που, στα συγκεκριμένα ερωτήματα και με το δοσμένο σύνολο δεδομένων, οι δύο μέθοδοι φαίνεται να παρέχουν τα ίδια αποτελέσματα, εκτιμάται ότι η **τεχνική DTW** μπορεί να ξεπεράσει την MEI καθώς αυξάνεται η πολυπλοκότητα των χαρακτηριστικών που λαμβάνονται υπόψη. Αυτό συμβαίνει γιατί το DTW είναι πιο ευέλικτο και μπορεί να εκμεταλλευτεί περισσότερες πληροφορίες, όπως η θέση και το σχήμα του χεριού, βελτιώνοντας την ακρίβεια της ταξινόμησης. Ως εκ τούτου, με την εισαγωγή περισσότερων χαρακτηριστικών, όπως στην προσέγγιση (β), το DTW αναμένεται να υπερέχει έναντι της MEI στην αναγνώριση χειρονομιών.

3 Κώδικες

3.1 classifier_mei

```
clear; clc; close all;
%%
% Define the base folder containing all gesture datasets
baseFolder = pwd;

% Define subfolders for each gesture category and the number of sequences
gestureFolders = {'Clic', 'No', 'Rotate', 'StopGraspOk'};
numSeqPerFolder = [15, 14, 13, 15];

% Define the number of training sequences per gesture
numTrainSeq = 5;

% Initialize variables for storing training and testing data
trainingData = [];
trainingLabels = [];
testingData = [];
testingLabels = [];

%% Loop through each gesture folder (representing different classes)
for gestureIdx = 1:length(gestureFolders)
    folderName = gestureFolders{gestureIdx};
    numSeq = numSeqPerFolder(gestureIdx);

    % Loop through the sequences within each gesture folder
    for seqIdx = 1:numSeq
        % Set the folder for the current sequence
        seqFolder = fullfile(baseFolder, folderName, ['Seq' num2str(seqIdx)]);

        % List all .pnm files in the current sequence folder
        imageFiles = dir(fullfile(seqFolder, '*.pnm'));

        % Extract MEI for the current sequence
        mei = extractMEI(seqFolder, imageFiles);

        if seqIdx <= numTrainSeq
            % Store MEI and label for training data
            trainingData = [trainingData; mei(:)'];  % Flatten MEI to a vector
            trainingLabels = [trainingLabels; gestureIdx];
        else
            % Store MEI and label for testing data
            testingData = [testingData; mei(:)'];
            testingLabels = [testingLabels; gestureIdx];
        end
    end
end

%% Implement k-NN classifier
k = 3;
mdl = fitcknn(trainingData, trainingLabels, 'NumNeighbors', k);

% Predict using the k-NN model
predictedLabels = predict(mdl, testingData);

%% Confusion Matrix
% Generate Confusion Matrix
confMatrix = confusionmat(testingLabels, predictedLabels);

% Display Confusion Matrix
```

```
disp('Confusion Matrix:');
disp(confMatrix);

% Confusion Matrix Heatmap
figure;
heatmap(confMatrix, 'Title', 'Confusion Matrix', 'XLabel', 'Predicted Class',
'YLabel', 'True Class');

% Display Total Accuracy
accuracy = sum(diag(confMatrix)) / sum(confMatrix(:));
disp(['Overall Accuracy: ', num2str(accuracy * 100), '%']);

% Display Per-Class Accuracy
classAccuracy = diag(confMatrix) ./ sum(confMatrix, 2);
disp('Per-Class Accuracy:');
disp(classAccuracy);

%% Testing with Different k Values
ks = 1:10;
accuracies = zeros(length(ks), 1);

% Initialize cell array to store confusion matrices
confMatrices = cell(length(ks), 1);

for i = 1:length(ks)
    % Train k-NN model with current k
    mdl = fitcknn(trainingData, trainingLabels, 'NumNeighbors', ks(i));
    % Predict using the k-NN model
    predictedLabels = predict(mdl, testingData);
    % Generate confusion matrix
    confMatrix = confusionmat(testingLabels, predictedLabels);
    % Store confusion matrix for current k
    confMatrices{i} = confMatrix;
    % Calculate accuracy
    accuracies(i) = sum(diag(confMatrix)) / sum(confMatrix(:));
end

% Plot k vs accuracy
figure;
plot(ks, accuracies, '-o');
xlabel('k');
ylabel('Accuracy');
title('k-NN Accuracy for Different k Values');

% Display Confusion Matrices
for i = 1:length(ks)
    figure;
    heatmap(confMatrices{i}, 'Title', ['Confusion Matrix for k = ' num2str(ks(i))],
...
            'XLabel', 'Predicted Class', 'YLabel', 'True Class', ...
            'XDisplayLabels', gestureFolders, 'YDisplayLabels', gestureFolders);
end
function mei = extractMEI(seqFolder, imageFiles)
    % Initialize MEI
    img = imread(fullfile(seqFolder, imageFiles(1).name));
    mei = zeros(size(img));

    % Loop through all images in the sequence
    for i = 2:length(imageFiles)
        % Read current and previous images
        imgPrev = imread(fullfile(seqFolder, imageFiles(i-1).name));
        imgCurr = imread(fullfile(seqFolder, imageFiles(i).name));

        % Calculate the difference image (binary)
```

```
        diffImg = abs(double(imgCurr) - double(imgPrev)) > 39; % Threshold for motion

        % Update MEI (aggregate motion)
        mei = mei + diffImg;
    end

    % Normalize MEI and convert to double
    mei = mat2gray(mei);  % Scale MEI to 0-1
    mei = double(mei);

end
```

3.2 classifier_dtw

```
clear; clc; close all;
featureMode = 'position_and_shape'; % Options: 'position_only', 'position_and_shape'
%% Define the base folder containing all gesture datasets
baseFolder = pwd;

% Define subfolders for each gesture category and the number of sequences
gestureFolders = {'Clic', 'No', 'Rotate', 'StopGraspOk'};
numSeqPerFolder = [15, 14, 13, 15];

% Define the number of training sequences per gesture
numTrainSeq = 5;

% Initialize variables for storing training and testing data
trainingData = [];
trainingLabels = [];
testingData = [];
testingLabels = [];

%% Extract features from each gesture folder
for gestureIdx = 1:length(gestureFolders)
    folderName = gestureFolders{gestureIdx};
    numSeq = numSeqPerFolder(gestureIdx);

    % Loop through the sequences within each gesture folder
    for seqIdx = 1:numSeq
        % Set the folder for the current sequence
        seqFolder = fullfile(baseFolder, folderName, ['Seq' num2str(seqIdx)]);

        % List all .pnm files in the current sequence folder
        imageFiles = dir(fullfile(seqFolder, '*.pnm'));

        % Extract features (position and shape) for the current sequence
        features = extractFeatures(seqFolder, imageFiles,featureMode);

        if seqIdx <= numTrainSeq
            % Store features and label for training data
            trainingData = [trainingData; {features}];
            trainingLabels = [trainingLabels; gestureIdx];
        else
            % Store features and label for testing data
            testingData = [testingData; {features}];
            testingLabels = [testingLabels; gestureIdx];
        end
    end
end

%% Testing with Different k Values
ks = 1:10;
accuracies = zeros(length(ks), 1);
```

```
% Initialize cell array to store confusion matrices
confMatrices = cell(length(ks), 1);

for i = 1:length(ks)
    predictedLabels = zeros(size(testingLabels));

    % Loop through each test sequence
    for j = 1:length(testingLabels)
        % Get the current test sample
        testSample = testingData{j};

        % Calculate DTW distance between the test sample and all training samples
        dtwDistances = zeros(1, length(trainingLabels));
        for k = 1:length(trainingLabels)
            dtwDistances(k) = dtwDistance(testSample, trainingData{k});
        end

        % Find the k nearest neighbors based on DTW distance
        [~, nnIndices] = sort(dtwDistances, 'ascend');
        nnIndices = nnIndices(1:ks(i));

        % Predict the label based on majority voting
        predictedLabels(j) = mode(trainingLabels(nnIndices));
    end

    % Generate confusion matrix
    confMatrix = confusionmat(testingLabels, predictedLabels);
    % Store confusion matrix for current k
    confMatrices{i} = confMatrix;
    % Calculate accuracy
    accuracies(i) = sum(diag(confMatrix)) / sum(confMatrix(:));
end
%%
ConfMatrix = confMatrices{3};
% Display Confusion Matrix for k=3
disp('Confusion Matrix:');
disp(ConfMatrix);
% Display Total Accuracy for k=3
accuracy = sum(diag(ConfMatrix)) / sum(ConfMatrix(:));
disp(['Overall Accuracy: ', num2str(accuracy * 100), '%']);
% Display Per-Class Accuracy for k=3
classAccuracy = diag(ConfMatrix) ./ sum(ConfMatrix, 2);
disp('Per-Class Accuracy:');
disp(classAccuracy);
% Confusion Matrix Heatmap
figure;
heatmap(ConfMatrix, 'Title', 'Confusion Matrix', 'XLabel', 'Predicted Class',
'YLabel', 'True Class',...
    'XDisplayLabels', gestureFolders, 'YDisplayLabels', gestureFolders);


% Plot k vs accuracy
figure;
plot(ks, accuracies, '-o');
xlabel('k');
ylabel('Accuracy');
title('k-NN Accuracy for Different k Values with DTW');

% Display Confusion Matrices
figure;
for i = 1:length(ks)
    subplot(2, 5, i);
```

```
    heatmap(confMatrices{i}, 'Title', ['Confusion Matrix for k = ' num2str(ks(i))],
...
            'XLabel', 'Predicted Class', 'YLabel', 'True Class', ...
            'XDisplayLabels', gestureFolders, 'YDisplayLabels', gestureFolders);
end

%% Function to Extract Features
function features = extractFeatures(seqFolder, imageFiles,featureMode)
    features = [];

    for i = 1:length(imageFiles)
        img = imread(fullfile(seqFolder, imageFiles(i).name));

        % Reshape image into a 2D array where each row is a pixel, and columns are the
RGB values
        reshapedImg = reshape(img, [], 3);

        % Estimate the background color as the most common color in the image
        backgroundColor = mode(double(reshapedImg), 1);

        % Calculate the Euclidean distance of each pixel to the background color
        distanceFromBackground = sqrt(sum((double(reshapedImg) - backgroundColor).^2,
2));

        % Reshape the distance array back to the image dimensions
        distanceImg = reshape(distanceFromBackground, size(img, 1), size(img, 2));

        % Threshold the distance image to create a binary mask
        handMask = distanceImg > 90;

        % Get the position of the hand (center of mass)
        [rows, cols] = find(handMask);
        if isempty(rows)
            continue;
        end

        centerX = mean(cols);
        centerY = mean(rows);

        % Normalize the features (position and shape)
        normalizedCenterX = centerX / 58;
        normalizedCenterY = centerY / 62;

        switch featureMode
            case 'position_only'
                %% Step (a) - Hand represented as position (i, j) only
                features = [features; normalizedCenterX, normalizedCenterY];

            case 'position_and_shape'
                %% Step (b) - Hand represented as position (i, j) and shape (bounding
box dimensions)
                % Get the bounding box of the hand
                minRow = min(rows);
                maxRow = max(rows);
                minCol = min(cols);
                maxCol = max(cols);

                width = maxCol - minCol;
                height = maxRow - minRow;

                % Normalize the shape features (width and height)
                normalizedWidth = width / 58;
                normalizedHeight = height / 62;

                % Combine position and shape features
```

```
                features = [features; normalizedCenterX, normalizedCenterY,
normalizedWidth, normalizedHeight];
            end
        end
    end

%% Function to Compute DTW Distance
function dist = dtwDistance(seq1, seq2)
    % Compute DTW distance between two sequences
    n = size(seq1, 1);
    m = size(seq2, 1);

    % Initialize cost matrix
    costMatrix = inf(n+1, m+1);
    costMatrix(1, 1) = 0;

    % Compute cost matrix
    for i = 1:n
        for j = 1:m
            cost = sum((seq1(i, :) - seq2(j, :)).^2);
            costMatrix(i+1, j+1) = cost + min([costMatrix(i, j+1), costMatrix(i+1, j),
costMatrix(i, j)]);
        end
    end

    % DTW distance is the value in the bottom-right corner of the cost matrix
    dist = sqrt(costMatrix(n+1, m+1));
end
```