

PROJECT

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΟΝΟΜΑ:

ΣΑΒΒΑΣ

ΕΠΩΝΥΜΟ:

ΡΟΣΤΑΝΤΗΣ

ΑΡ.ΜΗΤΡΩΟΥ:

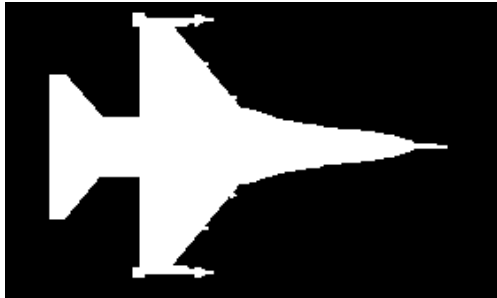
1115201000149

ΣΚΟΠΟΣ:

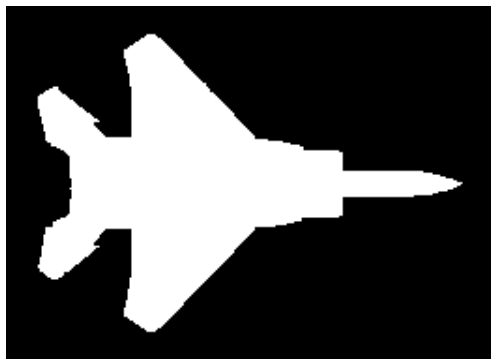
Στην Άσκηση αυτή ζητείται να ταξινομήσουμε αυτόματα μια σειρά από δισδιάστατες μορφές σε μία από τρεις γνωστές κλάσεις. Για το σκοπό αυτό δίνονται τα δεδομένα από 15 αναγνωρισμένες μορφές (*labeled_samples*) 5 για κάθε τάξη. Ως ουσιώδη χαρακτηριστικά των μορφών αυτών που θα επιλέγουν, θα χρησιμοποιηθούν τρεις από τις ροπές του Hue (Αμετάβλητες Ροπές), θα χρησιμοποιηθούν ο ταξινομητής ελάχιστης Ευκλείδειας απόστασης και ο ταξινομητής 3-NN. Τα τυχαία δείγματα (test-sets) θα τα παράγουμε από την συνάρτησης ,γεννήτριας τυχαίων samples. Ύστερα από την εκπαίδευση του ταξινομητή θα χρησιμοποιηθούν τα τυχαία δείγματα για τον έλεγχο της ορθότητας των ταξινομητών μας .

ΟΙ ΤΡΕΙΣ ΚΛΑΣΕΙΣ ΜΑΣ ΕΧΟΥΝΕ ΤΑ ΕΞΗΣ ΠΡΟΤΥΠΑ

1_{HC} ΚΛΑΣΗ:



2_{HC} ΚΛΑΣΗ:



3_{HC} ΚΛΑΣΗ:



ΤΑ ΔΕΔΟΜΕΝΑ ΕΚΠΑΙΔΕΥΣΗΣ

1_{HC} ΚΛΑΣΗ:



2_{HC} ΚΛΑΣΗ:



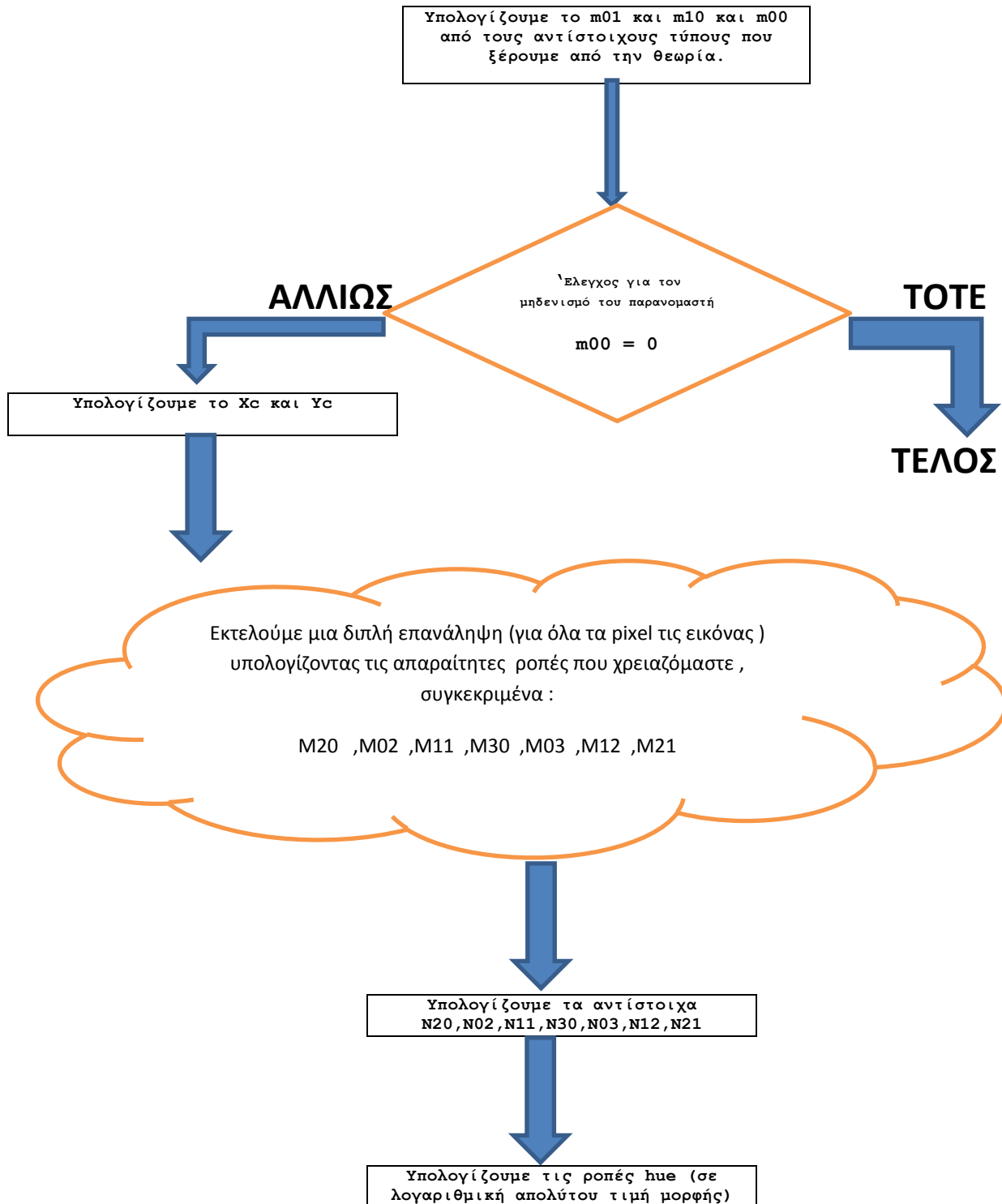
3_{HC} ΚΛΑΣΗ:




- ❖ Ο λόγος που θα χρησιμοποιήσουμε τις ροπές HUE είναι ότι τόσο στα δεδομένα εκπαίδευσης όσο και στα τυχαία δεδομένα της γεννήτριας με τις συγκεκριμένες ροπές δεν μας αφορά το μέγεθος ,η περιστροφή και η αντίθεση των εικόνων . Έτσι τα πράγματα απλοποιούνται στην ταξινόμηση τους.

ΡΟΠΕΣ ΗΥΕ

➤ ΥΠΟΛΟΓΙΣΜΟΣ ΡΟΠΩΝ




 **ΣΧΟΛΙΑ:** Στο πρόγραμμα αυτό έχουμε μια συνάρτηση που υπολογίζει της ροπές h_{ue} . Αρχικά υπολογίζουμε τα m_{10} και m_{01} από τους αντίστοιχους τύπους. Υπολογίζουμε τα κέντρα X_c, Y_c και υπολογίζουμε τα απαρρέτητα M για τον υπολογισμό των 7 ροπών, υπολογίζοντας βέβαια την απόλυτη τιμή του λογαρίθμου. Τέλος συγκρίνουμε τα αποτελέσματα με το δοσμένο αρχείο για τον έλεγχο της ορθότητας.

➤ ΔΙΑΒΑΣΜΑ ΕΙΚΟΝΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΜΟΣ ΡΟΠΩΝ ΗΥΕ

- Μια εικόνα στο Matlab διαβάζεται με την συνάρτηση `imread`.
- Και με την συνάρτηση `hue` που δημιουργήσαμε υπολογίζουμε τις ροπές .

```
pic1_1=imread('S1_1.tif');  
hue1_1=hue(pic1_1);  
pic1_2=imread('S1_2.tif');  
hue1_2=hue(pic1_2);  
pic1_3=imread('S1_3.tif');  
hue1_3=hue(pic1_3);  
pic1_4=imread('S1_4.tif');  
hue1_4=hue(pic1_4);  
pic1_5=imread('S1_5.tif');  
hue1_5=hue(pic1_5);  
pic2_1=imread('S2_1.tif');  
hue2_1=hue(pic2_1);  
pic2_2=imread('S2_2.tif');  
hue2_2=hue(pic2_2);  
pic2_3=imread('S2_3.tif');  
hue2_3=hue(pic2_3);  
pic2_4=imread('S2_4.tif');  
hue2_4=hue(pic2_4);  
pic2_5=imread('S2_5.tif');  
hue2_5=hue(pic2_5);  
pic3_1=imread('S3_1.tif');  
hue3_1=hue(pic3_1);  
pic3_2=imread('S3_2.tif');  
hue3_2=hue(pic3_2);  
pic3_3=imread('S3_3.tif');  
hue3_3=hue(pic3_3);  
pic3_4=imread('S3_4.tif');  
hue3_4=hue(pic3_4);  
pic3_5=imread('S3_5.tif');  
hue3_5=hue(pic3_5)
```

 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι τα αποτελέσματα μας είναι τα ίδια με του δοσμένου αρχείου.

<u>ΡΟΠΕΣ</u> <u>ΕΙΚΟΝΕΣ</u>	ΡΟΠΗ_1	ΡΟΠΗ_2	ΡΟΠΗ_3	ΡΟΠΗ_4	ΡΟΠΗ_5	ΡΟΠΗ_6	ΡΟΠΗ_7
S1_1	-1,3478	-4,4825	-5.2461	-6.5298	-12.4178	-8.7711	-17.1818
S1_2	-1,3473	-4,4717	-5.1650	-6.4461	-12.2517	-8.6821	-16.8590
S1_3	-1,3478	-4,4636	-5.1683	-6.4450	-12.2517	-8.6770	-16.7141
S1_4	-1,3475	-4,4851	-5.2453	-6.5312	-12.4194	-8.7738	-17.3504
S1_5	-1,3481	-4,4768	-5.1700	-6.4512	-12.2619	-8.6899	-17.2102
S2_1	-1,4504	-4,9286	-5.0840	-6.8624	-12.8356	-9.3267	-18.3461
S2_2	-1,4512	-4,9373	-5.0881	-6.8758	-12.8577	-9.3444	-19.9007
S2_3	-1,4500	-4,9166	-5.0899	-6.8495	-12.8192	-9.3078	-17.9940
S2_4	-1,4504	-4,9286	-5.0840	-6.8624	-12.8356	-9.3267	-18.3461
S2_5	-1,4500	-5,7013	-5.0899	-6.8495	-12.8192	-9.3078	-17.9940
S3_1	-1,1752	-5,7949	-7.8038	-9.9350	-18.9499	-12.8280	-19.4925
S3_2	-1,1775	-5,7949	-7.6641	-9.6131	-18.2616	-12.5150	-20.2194
S3_3	-1,1778	-5,7971	-7.6845	-9.6364	-18.2972	-12.5356	-21.9966
S3_4	-1,1762	-5,7349	-7.8481	-9.8948	-18.8985	-12.8050	-19.4955
S3_5	-1,1778	-5,7971	-7.6845	-9.6364	-18.2972	-12.5356	-21.9966

ΚΕΝΤΡΟΕΙΔΗ

➤ ΥΠΟΛΟΓΙΣΜΟΣ ΚΕΝΤΡΟΕΙΔΩΝ

Αφού έχουμε διαβάσει και τις 15 εικόνες εκπαίδευσης, και έχουμε υπολογίσει και τις ροπές τους διαχωρίζουμε τις τρεις πρώτες. Συγκεκριμένα:

```
ropes1_1=hue1_1(1:3);  
ropes1_2=hue1_2(1:3);  
ropes1_3=hue1_3(1:3);  
ropes1_4=hue1_4(1:3);  
ropes1_5=hue1_5(1:3);  
  
ropes2_1=hue2_1(1:3);  
ropes2_2=hue2_2(1:3);  
ropes2_3=hue2_3(1:3);  
ropes2_4=hue2_4(1:3);  
ropes2_5=hue2_5(1:3);  
  
ropes3_1=hue3_1(1:3);  
ropes3_2=hue3_2(1:3);  
ropes3_3=hue3_3(1:3);  
ropes3_4=hue3_4(1:3);  
ropes3_5=hue3_5(1:3);
```

Και υπολογίζουμε τα κεντροειδή αθροίζοντας τις πέντε ροπές κάθε κλάσης και διαιρώντας με το πλήθος τους. Παίρνουμε δηλαδή των μέσο όρο.

```
kedroeides_1=(ropes1_1+ropes1_2+ropes1_3+ropes1_4+ropes1_5)./5;  
kedroeides_2=(ropes2_1+ropes2_2+ropes2_3+ropes2_4+ropes2_5)./5;  
kedroeides_3=(ropes3_1+ropes3_2+ropes3_3+ropes3_4+ropes3_5)./5;
```

ΣΧΟΛΙΑ:

Παίρνουμε τις τρεις πρώτες ροπές, γιατί είναι αρκετές για την άσκηση , και ύστερα υπολογίζουμε τα κέντρα για τις τρεις κλάσεις μας.

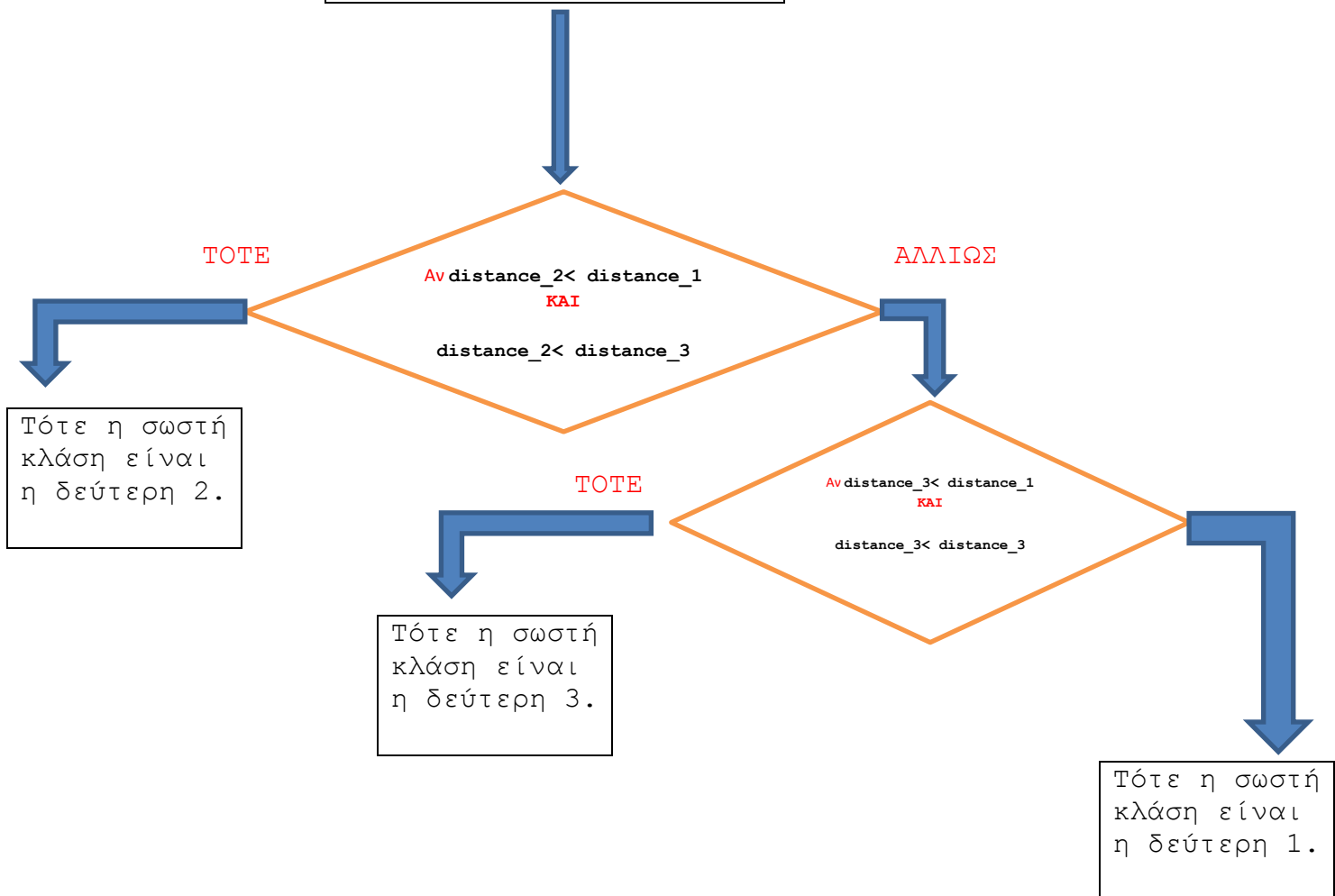
KENTPO 1	KENTPO 2	KENTPO 3
-1.3477 , -4.4759 , -5.1989	1.4504 , -4.9255 , -5.0872	-1.1769 , -5.7651 , -7.7370

ΤΑΞΙΝΟΜΗΤΗΣ ΕΛΑΧ.ΑΠΟΣΤΑΣΗΣ(ΕΥΚ)

➤ ΔΗΜΙΟΥΡΓΙΑ ΤΑΞΙΝΟΜΗΤΗ

Παίρνουμε τις τρεις πρώτες ροπές Hue από τις επτά ,χρησιμοποιώντας την συνάρτηση Hue που κατασκευάσαμε.
x,y,z

Υπολογίζουμε τρεις ελάχιστες αποστάσεις. Για κάθε απόσταση παίρνουμε την ευκλείδεια απόσταση των ροπών από τα κεντροειδή κάθε κλάσης.
distance_1, distance_2, distance_3



ΣΧΟΛΙΑ:

Σε αυτό το σημείο δημιουργήσαμε τον ταξινομητή ευκλείδειας ελάχιστης απόστασης τον οποίο ύστερα θα τον εκπαιδεύσουμε (training_set) για τον υπολογισμό των τυχαίων δειγμάτων. Η συνάρτηση δέχεται ως όρισμα την εικόνα και τα υπολογισμένα κέντρα της κάθε κλάσης και υπολογίζει την ελάχιστη απόσταση των ροπών της εικόνας και επιστρέφει τον αριθμό(1,2,3) της κλάσης που ταξινομήθηκε .

➤ ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΤΑΞΙΝΟΜΗΤΗ


- Για την εκπαίδευση του ταξινομητή απλά του περνάμε κάθε μια εικόνα εκπαίδευσης και τα τρία κεντροειδή. Στο τέλος απλά ελέγχουμε αν η κλάση που μας βγάζει η συνάρτηση είναι ίδια με αυτή που ανήκει η κάθε γνωστή εικόνα των δειγμάτων εκπαίδευσης .

ΣΥΓΚΕΚΡΙΜΕΝΑ:

```
class1_1=Euclidean_Classifier(pic1_1,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class1_2=Euclidean_Classifier(pic1_2,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class1_3=Euclidean_Classifier(pic1_3,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class1_4=Euclidean_Classifier(pic1_4,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class1_5=Euclidean_Classifier(pic1_5,kedroeides_1,kedroeides_2,kedroe  
ides_3);
```

```
class2_1=Euclidean_Classifier(pic2_1,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class2_2=Euclidean_Classifier(pic2_2,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class2_3=Euclidean_Classifier(pic2_3,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class2_4=Euclidean_Classifier(pic2_4,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class2_5=Euclidean_Classifier(pic2_5,kedroeides_1,kedroeides_2,kedroe  
ides_3);
```

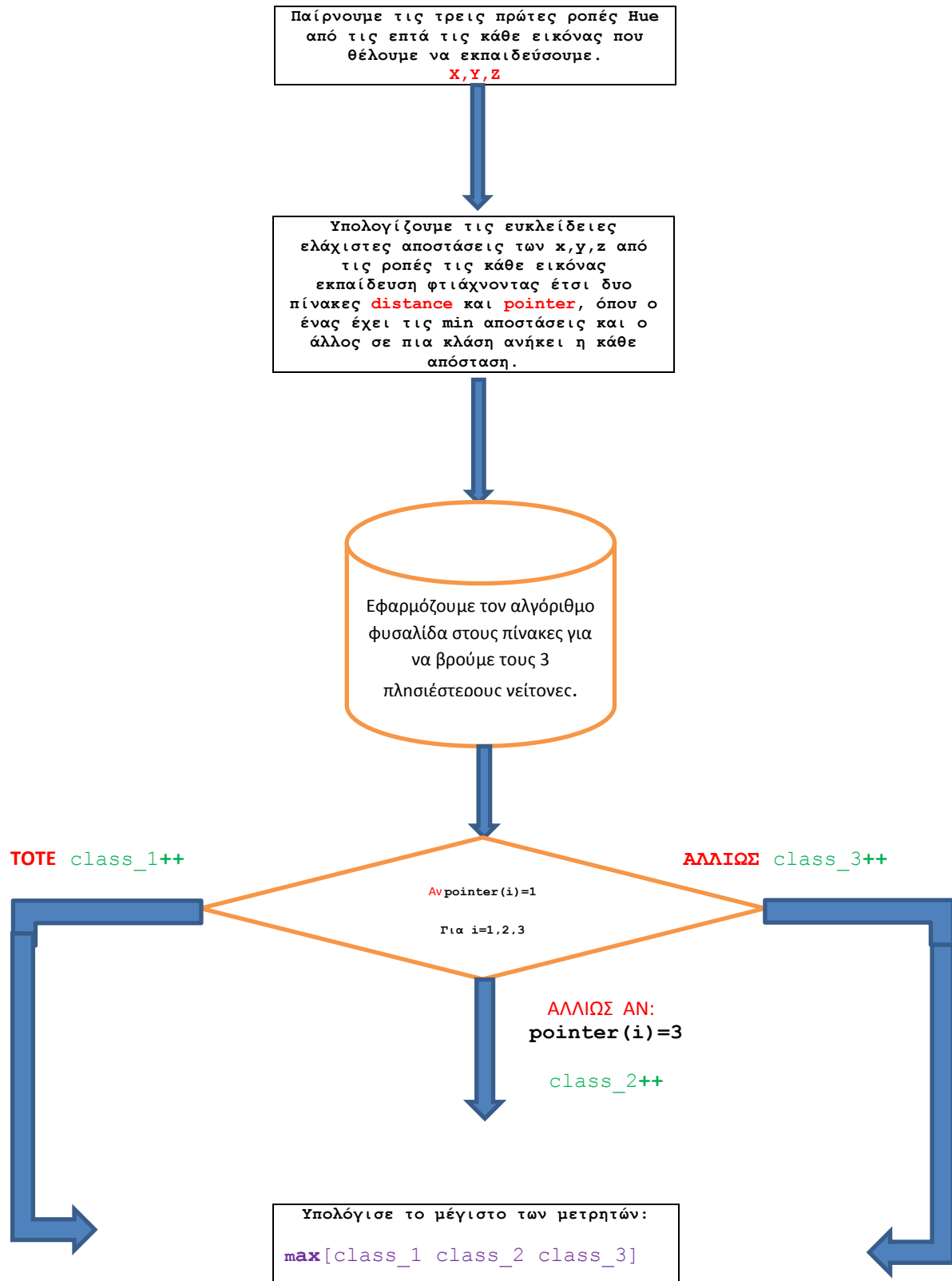
```
class3_1=Euclidean_Classifier(pic3_1,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class3_2=Euclidean_Classifier(pic3_2,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class3_3=Euclidean_Classifier(pic3_3,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class3_4=Euclidean_Classifier(pic3_4,kedroeides_1,kedroeides_2,kedroe  
ides_3);  
class3_5=Euclidean_Classifier(pic3_5,kedroeides_1,kedroeides_2,kedroe  
ides_3);
```


 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι ο ταξινομητής μας εκπαιδεύτηκε σωστά, ταξινόμησε όλες τις εικόνες μας σωστά.

ΕΙΚΟΝΕΣ	ΑΝΗΚΕΙ ΣΤΗΝ ΚΛΑΣΗ	ΤΑΞΙΝΟΜΗΘΗΚΕ ΣΤΗΝ ΚΛΑΣΗ
S1_1	<u>1</u>	<u>1</u>
S1_2	<u>1</u>	<u>1</u>
S1_3	<u>1</u>	<u>1</u>
S1_4	<u>1</u>	<u>1</u>
S1_5	<u>1</u>	<u>1</u>
S2_1	<u>2</u>	<u>2</u>
S2_2	<u>2</u>	<u>2</u>
S2_3	<u>2</u>	<u>2</u>
S2_4	<u>2</u>	<u>2</u>
S2_5	<u>2</u>	<u>2</u>
S3_1	<u>3</u>	<u>3</u>
S3_2	<u>3</u>	<u>3</u>
S3_3	<u>3</u>	<u>3</u>
S3_4	<u>3</u>	<u>3</u>
S3_5	<u>3</u>	<u>3</u>

ΤΑΞΙΝΟΜΗΤΗΣ 3-NN

➤ ΔΗΜΙΟΥΡΓΙΑ ΤΑΞΙΝΟΜΗΤΗ



 **ΣΧΟΛΙΑ:** Στη συγκεκριμένη συνάρτηση γίνεται ταξινόμηση μέσω του ταξινομητή 3-NN. Ύστερα υπολογίζουμε τις ευκλείδειες ελάχιστες αποστάσεις της κάθε τυχαίας εικόνας (`test_test`) από και τις 15 εικόνες εκπαίδευσης (`training_set`) και αποθηκεύουμε τα αποτελέσματα στον πίνακα `distance` και στον πίνακα `pointer` τις κλάσεις διαχωρισμένες. Γίνεται μετά ταξινόμηση στον πίνακα `distance` και παίρνουμε τους 3 πιο κοντινούς γείτονες η οποίοι βρίσκονται πλέον στις 3 πρώτες θέσεις του πίνακα. Τέλος με την χρήση 3 μετρητών μετράμε τον ταξινομημένο πίνακα `pointer` και βρίσκουμε των μεγαλύτερο μετρητή ο οποίος θα αντιπροσωπεύει την σωστή κλάση.

➤ ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΤΑΞΙΝΟΜΗΤΗ


- Για την εκπαίδευση του ταξινομητή απλά του περνάμε τις ροπές τις κάθε εικόνας εκπαίδευσης καθώς και των πίνακα που έχει όλες τις ροπές όλων των δειγμάτων εκπαίδευσης . Στο τέλος απλά ελέγχουμε αν η κλάση που μας βγάζει η συνάρτηση είναι ίδια με αυτή που ανήκει η κάθε γνωστή εικόνα των δειγμάτων εκπαίδευσης .

ΣΥΓΚΕΚΡΙΜΕΝΑ:

```
class1_1=NN_Classifier(hue1_1,pinakas_3_ropwn);  
class1_2=NN_Classifier(hue1_2,pinakas_3_ropwn);  
class1_3=NN_Classifier(hue1_3,pinakas_3_ropwn);  
class1_4=NN_Classifier(hue1_4,pinakas_3_ropwn);  
class1_5=NN_Classifier(hue1_5,pinakas_3_ropwn);
```

```
class2_1=NN_Classifier(hue2_1,pinakas_3_ropwn);  
class2_2=NN_Classifier(hue2_2,pinakas_3_ropwn);  
class2_3=NN_Classifier(hue2_3,pinakas_3_ropwn);  
class2_4=NN_Classifier(hue2_4,pinakas_3_ropwn);  
class2_5=NN_Classifier(hue2_5,pinakas_3_ropwn);
```

```
class3_1=NN_Classifier(hue3_1,pinakas_3_ropwn);  
class3_2=NN_Classifier(hue3_2,pinakas_3_ropwn);  
class3_3=NN_Classifier(hue3_3,pinakas_3_ropwn);  
class3_4=NN_Classifier(hue3_4,pinakas_3_ropwn);  
class3_5=NN_Classifier(hue3_5,pinakas_3_ropwn);
```

 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι ο ταξινομητής μας εκπαιδεύτηκε σωστά, ταξινόμησε όλες τις εικόνες μας σωστά.

ΕΙΚΟΝΕΣ	ΑΝΗΚΕΙ ΣΤΗΝ ΚΛΑΣΗ	ΤΑΞΙΝΟΜΗΘΗΚΕ ΣΤΗΝ ΚΛΑΣΗ
S1_1	<u>1</u>	<u>1</u>
S1_2	<u>1</u>	<u>1</u>
S1_3	<u>1</u>	<u>1</u>
S1_4	<u>1</u>	<u>1</u>
S1_5	<u>1</u>	<u>1</u>
S2_1	<u>2</u>	<u>2</u>
S2_2	<u>2</u>	<u>2</u>
S2_3	<u>2</u>	<u>2</u>
S2_4	<u>2</u>	<u>2</u>
S2_5	<u>2</u>	<u>2</u>
S3_1	<u>3</u>	<u>3</u>
S3_2	<u>3</u>	<u>3</u>
S3_3	<u>3</u>	<u>3</u>
S3_4	<u>3</u>	<u>3</u>
S3_5	<u>3</u>	<u>3</u>

ΕΙΣΟΔΟΣ ΤΥΧΑΙΩΝ ΔΕΙΓΜΑΤΩΝ ΣΤΟΥΣ ΤΑΞΙΝΟΜΗΤΕΣ

- ❖ Πέρα των δειγμάτων εκπαίδευσης πρέπει οι ταξινομητές μας να είναι σε θέση να ταξινομούν οποιαδήποτε τυχαία εικόνα στην σωστή κλάση.
- ❖ Με τη βοήθεια μιας γεννήτριας τυχαίων εικόνων θα εισάγουμε στους ταξινομητές μας 40 τυχαίες εικόνες για ταξινόμηση , και τέλος θα αξιολογήσουμε τους ταξινομητές μας προς την απόδοση.


➤ ΕΥΚΛΕΙΔΕΙΟΣ ΤΑΞΙΝΟΜΗΤΗΣ

Οι τυχαίες εικόνες προς ταξινόμηση:



ΕΙΚΟΝΕΣ	ΑΝΗΚΕΙ ΣΤΗΝ ΚΛΑΣΗ	ΤΑΞΙΝΟΜΗΘΗΚΕ ΣΤΗΝ ΚΛΑΣΗ
<u>1^η</u>	3	3
<u>2^η</u>	2	2
<u>3^η</u>	2	2
<u>4^η</u>	1	1
<u>5^η</u>	1	1
<u>6^η</u>	2	2
<u>7^η</u>	3	3
<u>8^η</u>	2	2
<u>9^η</u>	1	1
<u>10^η</u>	3	3
<u>11^η</u>	2	2
<u>12^η</u>	2	2
<u>13^η</u>	1	1
<u>14^η</u>	1	1
<u>15^η</u>	1	1
<u>16^η</u>	3	3
<u>17^η</u>	3	3
<u>18^η</u>	3	3
<u>19^η</u>	1	1
<u>20^η</u>	2	2
<u>21^η</u>	1	1
<u>22^η</u>	3	3
<u>23^η</u>	2	2
<u>24^η</u>	3	3
<u>25^η</u>	2	2
<u>26^η</u>	2	2
<u>27^η</u>	3	3
<u>28^η</u>	3	3
<u>29^η</u>	3	3

<u>30^η</u>	1	1
<u>31^η</u>	2	2
<u>32^η</u>	1	1
<u>33^η</u>	1	1
<u>34^η</u>	2	2
<u>35^η</u>	1	1
<u>36^η</u>	2	2
<u>37^η</u>	3	3
<u>38^η</u>	2	2
<u>39^η</u>	2	2
<u>40^η</u>	2	2

 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι ο ταξινομητής μας ταξινόμησε σωστά όλες τις τυχαίες εικόνες της γεννήτριας μας .


➤ 3-NN ΤΑΞΙΝΟΜΗΤΗΣ

➤ Οι τυχαίες εικόνες προς ταξινόμηση:



ΕΙΚΟΝΕΣ	ΑΝΗΚΕΙ ΣΤΗΝ ΚΛΑΣΗ	ΤΑΞΙΝΟΜΗΘΗΚΕ ΣΤΗΝ ΚΛΑΣΗ
<u>1^η</u>	3	3
<u>2^η</u>	2	2
<u>3^η</u>	2	2
<u>4^η</u>	1	1
<u>5^η</u>	1	1
<u>6^η</u>	2	2
<u>7^η</u>	3	3
<u>8^η</u>	2	2
<u>9^η</u>	1	1
<u>10^η</u>	3	3
<u>11^η</u>	2	2
<u>12^η</u>	2	2
<u>13^η</u>	1	1
<u>14^η</u>	1	1
<u>15^η</u>	1	1
<u>16^η</u>	3	3
<u>17^η</u>	3	3
<u>18^η</u>	3	3
<u>19^η</u>	1	1
<u>20^η</u>	2	2
<u>21^η</u>	1	1
<u>22^η</u>	3	3
<u>23^η</u>	2	2
<u>24^η</u>	3	3
<u>25^η</u>	2	2
<u>26^η</u>	2	2
<u>27^η</u>	3	3
<u>28^η</u>	3	3
<u>29^η</u>	3	3

<u>30^η</u>	1	1
<u>31^η</u>	2	2
<u>32^η</u>	1	1
<u>33^η</u>	1	1
<u>34^η</u>	2	2
<u>35^η</u>	1	1
<u>36^η</u>	2	2
<u>37^η</u>	3	3
<u>38^η</u>	2	2
<u>39^η</u>	2	2
<u>40^η</u>	2	2

 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι ο ταξινομητής μας ταξινόμησε σωστά όλες τις τυχαίες εικόνες της γεννήτριας μας .

ΑΞΙΟΛΟΓΗΣΗ ΤΩΝ ΕΠΙΔΟΣΕΩΝ ΤΩΝ ΤΑΞΙΝΟΜΗΤΩΝ

✚ Σε αυτό το σημείο πρέπει να δούμε πόσο αποδοτικοί είναι οι ταξινομητές μας . Στο πλαίσιο της άσκησης έχουμε 40 δείγματα για ταξινόμηση (test_set) .

✚ Για να υπολογίσουμε την απόδοση τους θα συγκρίνουμε της επιτυχίες που είχε ο ταξινομητής με την ταξινόμηση , με τις αποτυχίες του

- **Επιτυχία** : Επιτυχία είναι ο ταξινομητής να ταξινομήσει σωστά την εικόνα.
- **Αποτυχία** : Αποτυχία είναι ο ταξινομητής να μην ταξινομήσει σωστά την εικόνα.

➤ ΕΥΚΛΙΔΙΟΣ ΤΑΞΙΝΟΜΗΤΗΣ


- Στις 40 εικόνες είχαμε:
 - 40 επιτυχίες
 - 0 αποτυχίες

ΕΙΧΕ 100% ΕΠΙΤΥΧΙΑ

➤ 3-NN ΤΑΞΙΝΟΜΗΤΗΣ

- Στις 40 εικόνες είχαμε:
 - 40 επιτυχίες
 - 0 αποτυχίες

ΕΙΧΕ 100% ΕΠΙΤΥΧΙΑ

 **ΣΧΟΛΙΑ:** Παρατηρούμε ότι οι ταξινομητές μας ,τις 40 εικόνες για testing, τις ταξινόμησαν πλήρως σωστά . Είχαμε 100% απόδοση και στους δυο με μηδέν σφάλματα.

ΠΑΡΑΡΤΗΜΑ

ΚΩΔΙΚΕΣ ΣΕ MATLAB

➤ ΥΠΟΛΟΓΙΣΜΟΣ ΡΟΠΩΝ

ΠΡΟΓΡΑΜΜΑ ΥΠΟΛΟΓΙΣΜΟΥ:

```
• function [ ropes_f ] = hue( image )
• s=size(image);
• temp=sum(image);
• m00=sum(temp);
• m10=0;
• m01=0;
• for x=1:s(1)
•     for y=1:s(2)
•         m10=m10+x*image(x,y);
•         m01=m01+y*image(x,y);
•     end
• end

• if m00 ~= 0
•     Xc = m10 / m00;
•     Yc = m01 / m00;
• else
•     return;
• end

• M02=0;
• M20=0;
• M11=0;
• M12=0;
• M21=0;
• M30=0;
• M03=0;
• for x = 1 : s(1)
•     for y = 1 : s(2)
•         M20 = M20 + (x - Xc)^2 * image(x,y);
•         M02 = M02 + (y - Yc)^2 * image(x,y);
•         M11 = M11 + (x - Xc) * (y - Yc) * image(x,y);
•         M30 = M30 + (x - Xc)^3 * image(x,y);
•         M03 = M03 + (y - Yc)^3 * image(x,y);
•         M12 = M12 + (x - Xc) * (y - Yc)^2 * image(x,y);
•         M21 = M21 + (x - Xc)^2 * (y - Yc) * image(x,y);
•     end
• End

• N20 = M20 / (m00^2);
• N02 = M02 / (m00^2);
• N11 = M11 / (m00^2);
• N30 = M30 / (m00^2.5);
• N03 = M03 / (m00^2.5);
• N12 = M12 / (m00^2.5);
• N21 = M21 / (m00^2.5);

• ropes_f(1) =log(abs( N20 + N02));
• ropes_f(2) = log(abs((N20 - N02)^2 + 4 * (N11^2)));
• ropes_f(3) =log(abs((N30 - 3*N12)^2 + (3*N21 - N03)^2));
• ropes_f(4) = log(abs((N30 + N12)^2 + (N21 + N03)^2));
```

- `ropes_f(5) = log(abs((N30 - 3*N12) * (N30 + N12) * ((N30 + N12)^2 - 3 * (N21 + N03)^2) + (3*N21 - N03) * (N21 + N03) * (3*(N30 + N12)^2 - (N21 + N03)^2)));`
- `ropes_f(6) = log(abs((N20 - N02) * ((N30 + N12)^2 - (N21 + N03)^2) + 4 * N11 * (N30 + N12) * (N21 + N03)));`
- `ropes_f(7) = log(abs((3 * N21 - N03) * (N30 + N12) * ((N30 + N12)^2 - 3 * (N21 + N03)^2) + (N30 - 3 * N12) * (N21 + N03) * ((N21 + N03)^2 - 3 * (N30 + N12)^2)));`
- `end`

ΔΙΑΒΑΣΜΑ ΕΙΚΟΝΩΝ

ΠΡΟΓΡΑΜΜΑ :

- `pic1_1=imread('S1_1.tif');`
- `hue1_1=hue(pic1_1);`
- `pic1_2=imread('S1_2.tif');`
- `hue1_2=hue(pic1_2);`
- `pic1_3=imread('S1_3.tif');`
- `hue1_3=hue(pic1_3);`
- `pic1_4=imread('S1_4.tif');`
- `hue1_4=hue(pic1_4);`
- `pic1_5=imread('S1_5.tif');`
- `hue1_5=hue(pic1_5);`
- `pic2_1=imread('S2_1.tif');`
- `hue2_1=hue(pic2_1);`
- `pic2_2=imread('S2_2.tif');`
- `hue2_2=hue(pic2_2);`
- `pic2_3=imread('S2_3.tif');`
- `hue2_3=hue(pic2_3);`
- `pic2_4=imread('S2_4.tif');`
- `hue2_4=hue(pic2_4);`
- `pic2_5=imread('S2_5.tif');`
- `hue2_5=hue(pic2_5);`
- `pic3_1=imread('S3_1.tif');`
- `hue3_1=hue(pic3_1);`
- `pic3_2=imread('S3_2.tif');`
- `hue3_2=hue(pic3_2);`
- `pic3_3=imread('S3_3.tif');`
- `hue3_3=hue(pic3_3);`
- `pic3_4=imread('S3_4.tif');`
- `hue3_4=hue(pic3_4);`
- `pic3_5=imread('S3_5.tif');`
- `hue3_5=hue(pic3_5);`

➤ ΥΠΟΛΟΓΙΣΜΟΣ ΚΕΝΤΡΟΕΙΔΩΝ

ΠΡΟΓΡΑΜΜΑ ΥΠΟΛΟΓΙΣΜΟΥ:

- `ropes1_1=hue1_1(1:3);`
- `ropes1_2=hue1_2(1:3);`
- `ropes1_3=hue1_3(1:3);`
- `ropes1_4=hue1_4(1:3);`
- `ropes1_5=hue1_5(1:3);`

- `ropes2_1=hue2_1(1:3);`
- `ropes2_2=hue2_2(1:3);`
- `ropes2_3=hue2_3(1:3);`
- `ropes2_4=hue2_4(1:3);`
- `ropes2_5=hue2_5(1:3);`

- `ropes3_1=hue3_1(1:3);`
- `ropes3_2=hue3_2(1:3);`
- `ropes3_3=hue3_3(1:3);`
- `ropes3_4=hue3_4(1:3);`
- `ropes3_5=hue3_5(1:3);`

- `kedroeides_1=[(ropes1_1+ropes1_2+ropes1_3+ropes1_4+ropes1_5)./5];`
- `kedroeides_2=[(ropes2_1+ropes2_2+ropes2_3+ropes2_4+ropes2_5)./5];`
- `kedroeides_3=[(ropes3_1+ropes3_2+ropes3_3+ropes3_4+ropes3_5)./5];`

➤ ΔΗΜΙΟΥΡΓΙΑ ΤΑΞΙΝΟΜΗΤΗ

ΠΡΟΓΡΑΜΜΑ:

- `function [class] = Euclidean_Classifier(image,kedroeides_1,kedroeides_2,kedroeides_3)`
- `hue_of_image=hue(image);`
- `x=hue_of_image(1);`
- `y=hue_of_image(2);`
- `z=hue_of_image(3);`
- `distance_1=sqrt((x-kedroeides_1(1))^2+(y-kedroeides_1(2))^2+(z-kedroeides_1(3))^2);`
- `distance_2=sqrt((x-kedroeides_2(1))^2+(y-kedroeides_2(2))^2+(z-kedroeides_2(3))^2);`
- `distance_3=sqrt((x-kedroeides_3(1))^2+(y-kedroeides_3(2))^2+(z-kedroeides_3(3))^2);`
- `if distance_2<distance_1 && distance_2<distance_3`
- `class=2;`
- `else if distance_3<distance_1 && distance_3<distance_2`
- `class=3;`
- `else`
- `class=1;`
- `end`
- `end`
- `end`

➤ ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΤΑΞΙΝΟΜΗΤΗ

ΠΡΟΓΡΑΜΜΑ:

- `class1_1=Euclidean_Classifier(pic1_1,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class1_2=Euclidean_Classifier(pic1_2,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class1_3=Euclidean_Classifier(pic1_3,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class1_4=Euclidean_Classifier(pic1_4,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class1_5=Euclidean_Classifier(pic1_5,kedroeides_1,kedroeides_2,kedroeides_3);`

- `class2_1=Euclidean_Classifier(pic2_1,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class2_2=Euclidean_Classifier(pic2_2,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class2_3=Euclidean_Classifier(pic2_3,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class2_4=Euclidean_Classifier(pic2_4,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class2_5=Euclidean_Classifier(pic2_5,kedroeides_1,kedroeides_2,kedroeides_3);`

- `class3_1=Euclidean_Classifier(pic3_1,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class3_2=Euclidean_Classifier(pic3_2,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class3_3=Euclidean_Classifier(pic3_3,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class3_4=Euclidean_Classifier(pic3_4,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class3_5=Euclidean_Classifier(pic3_5,kedroeides_1,kedroeides_2,kedroeides_3);`

✚ ΠΡΟΓΡΑΜΜΑ:

```

• function [ class ] = NN_Classifier( hue_image,hue_matrix )
• x=hue_image(1);
• y=hue_image(2);
• z=hue_image(3);
• distance(1)=sqrt((x-hue_matrix(1,1))^2+(y-hue_matrix(1,2))^2+(z-
hue_matrix(1,3))^2);
• pointer(1)=1;
• distance(2)=sqrt((x-hue_matrix(2,1))^2+(y-hue_matrix(2,2))^2+(z-
hue_matrix(2,3))^2);
• pointer(2)=1;
• distance(3)=sqrt((x-hue_matrix(3,1))^2+(y-hue_matrix(3,2))^2+(z-
hue_matrix(3,3))^2);
• pointer(3)=1;
• distance(4)=sqrt((x-hue_matrix(4,1))^2+(y-hue_matrix(4,2))^2+(z-
hue_matrix(4,3))^2);
• pointer(4)=1;
• distance(5)=sqrt((x-hue_matrix(5,1))^2+(y-hue_matrix(5,2))^2+(z-
hue_matrix(5,3))^2);
• pointer(5)=1;
• distance(6)=sqrt((x-hue_matrix(6,1))^2+(y-hue_matrix(6,2))^2+(z-
hue_matrix(6,3))^2);
• pointer(6)=2;
• distance(7)=sqrt((x-hue_matrix(7,1))^2+(y-hue_matrix(7,2))^2+(z-
hue_matrix(7,3))^2);
• pointer(7)=2;
• distance(8)=sqrt((x-hue_matrix(8,1))^2+(y-hue_matrix(8,2))^2+(z-
hue_matrix(8,3))^2);
• pointer(8)=2;
• distance(9)=sqrt((x-hue_matrix(9,1))^2+(y-hue_matrix(9,2))^2+(z-
hue_matrix(9,3))^2);
• pointer(9)=2;
• distance(10)=sqrt((x-hue_matrix(10,1))^2+(y-
hue_matrix(10,2))^2+(z-hue_matrix(10,3))^2);
• pointer(10)=2;
• distance(11)=sqrt((x-hue_matrix(11,1))^2+(y-
hue_matrix(11,2))^2+(z-hue_matrix(11,3))^2);
• pointer(11)=3;
• distance(12)=sqrt((x-hue_matrix(12,1))^2+(y-
hue_matrix(12,2))^2+(z-hue_matrix(12,3))^2);
• pointer(12)=3;
• distance(13)=sqrt((x-hue_matrix(13,1))^2+(y-
hue_matrix(13,2))^2+(z-hue_matrix(13,3))^2);
• pointer(13)=3;
• distance(14)=sqrt((x-hue_matrix(14,1))^2+(y-
hue_matrix(14,2))^2+(z-hue_matrix(14,3))^2);
• pointer(14)=3;
• distance(15)=sqrt((x-hue_matrix(15,1))^2+(y-
hue_matrix(15,2))^2+(z-hue_matrix(15,3))^2);
• pointer(15)=3;
• for i=2:15
•     for j=15:-1:i
•         if distance(j-1)>distance(j)
•             temp_dist=distance(j-1);
•             temp_pointer=pointer(j-1);
•             distance(j-1)=distance(j);
•             pointer(j-1)=pointer(j);
•             distance(j)=temp_dist;
•             pointer(j)=temp_pointer;
•         end
•     end
• end
• class_1=0;
• class_2=0;

```

```

• class_3=0;
• for i=1:3
•     if pointer(i)==1
•         class_1=class_1+1;
•     end
•     if pointer(i)==2
•         class_2=class_2+1;
•     end
•     if pointer(i)==3
•         class_3=class_3+1;
•     end
• end
• if class_1>class_2
•     if class_1>class_3
•         class=1;
•     else
•         class=3;
•     end
• else
•     if class_2>class_3
•         class=2;
•     else
•         class=3;
•     end
• end
• end

```

➤ ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΤΑΞΙΝΟΜΗΤΗ

ΠΡΟΓΡΑΜΜΑ:

```

• pinakas_3_ropwn=[ropes1_1 ;ropes1_2 ;ropes1_3
;ropes1_4 ;ropes1_5;ropes2_1; ropes2_2 ;ropes2_3
;ropes2_4 ;ropes2_5 ; ropes3_1 ;ropes3_2
;ropes3_3;ropes3_4 ;ropes3_5];
• class1_1=NN_Classifier(hue1_1,pinakas_3_ropwn);
• class1_2=NN_Classifier(hue1_2,pinakas_3_ropwn);
• class1_3=NN_Classifier(hue1_3,pinakas_3_ropwn);
• class1_4=NN_Classifier(hue1_4,pinakas_3_ropwn);
• class1_5=NN_Classifier(hue1_5,pinakas_3_ropwn);
• class2_1=NN_Classifier(hue2_1,pinakas_3_ropwn);
• class2_2=NN_Classifier(hue2_2,pinakas_3_ropwn);
• class2_3=NN_Classifier(hue2_3,pinakas_3_ropwn);
• class2_4=NN_Classifier(hue2_4,pinakas_3_ropwn);
• class2_5=NN_Classifier(hue2_5,pinakas_3_ropwn);
• class3_1=NN_Classifier(hue3_1,pinakas_3_ropwn);
• class3_2=NN_Classifier(hue3_2,pinakas_3_ropwn);
• class3_3=NN_Classifier(hue3_3,pinakas_3_ropwn);
• class3_4=NN_Classifier(hue3_4,pinakas_3_ropwn);
• class3_5=NN_Classifier(hue3_5,pinakas_3_ropwn);

```

➤ ΤΑΞΙΝΟΜΗΣΗ ΤΥΧΑΙΩΝ ΔΕΙΓΜΑΤΩΝ

- `pic0=imread('photo0.tif');`
- `pic0=double(pic0)/255;`
- `hue0=hue(pic0);`
- `pic1=imread('photo1.tif');`
- `pic1=double(pic1)/255;`
- `hue1=hue(pic1);`
- `pic2=imread('photo2.tif');`
- `pic2=double(pic2)/255;`
- `hue2=hue(pic2);`
- `pic3=imread('photo3.tif');`
- `pic3=double(pic3)/255;`
- `hue3=hue(pic3);`
- `pic4=imread('photo4.tif');`
- `pic4=double(pic4)/255;`
- `hue4=hue(pic4);`
- `pic5=imread('photo5.tif');`
- `pic5=double(pic5)/255;`
- `hue5=hue(pic5);`
- `pic6=imread('photo6.tif');`
- `pic6=double(pic6)/255;`
- `hue6=hue(pic6);`
- `pic7=imread('photo7.tif');`
- `pic7=double(pic7)/255;`
- `hue7=hue(pic7);`
- `pic8=imread('photo8.tif');`
- `pic8=double(pic8)/255;`
- `hue8=hue(pic8);`
- `pic9=imread('photo9.tif');`
- `pic9=double(pic9)/255;`
- `hue9=hue(pic9);`
- `%..... ΟΜΟΙΑ ΓΙΑ ΤΙΣ ΥΠΟΛΟΙΠΕΣ ΕΙΚΟΝΕΣ`
- `class0=Euclidean_Classifier(pic0,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class1=Euclidean_Classifier(pic1,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class2=Euclidean_Classifier(pic2,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class3=Euclidean_Classifier(pic3,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class4=Euclidean_Classifier(pic4,kedroeides_1,kedroeides_2,kedroeides_3);`

- `class5=Euclidean_Classifier(pic5,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class6=Euclidean_Classifier(pic6,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class7=Euclidean_Classifier(pic7,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class8=Euclidean_Classifier(pic8,kedroeides_1,kedroeides_2,kedroeides_3);`
- `class9=Euclidean_Classifier(pic9,kedroeides_1,kedroeides_2,kedroeides_3);`

- `class0=NN_Classifier(hue0,pinakas_3_ropwn);`
- `class1=NN_Classifier(hue1,pinakas_3_ropwn);`
- `class2=NN_Classifier(hue2,pinakas_3_ropwn);`
- `class3=NN_Classifier(hue3,pinakas_3_ropwn);`
- `class4=NN_Classifier(hue4,pinakas_3_ropwn);`
- `class5=NN_Classifier(hue5,pinakas_3_ropwn);`
- `class6=NN_Classifier(hue6,pinakas_3_ropwn);`
- `class7=NN_Classifier(hue7,pinakas_3_ropwn);`
- `class8=NN_Classifier(hue8,pinakas_3_ropwn);`
- `class9=NN_Classifier(hue9,pinakas_3_ropwn);`