Πανεπιστήμιο Ιωαννίνων

Πολυτεχνική Σχολή

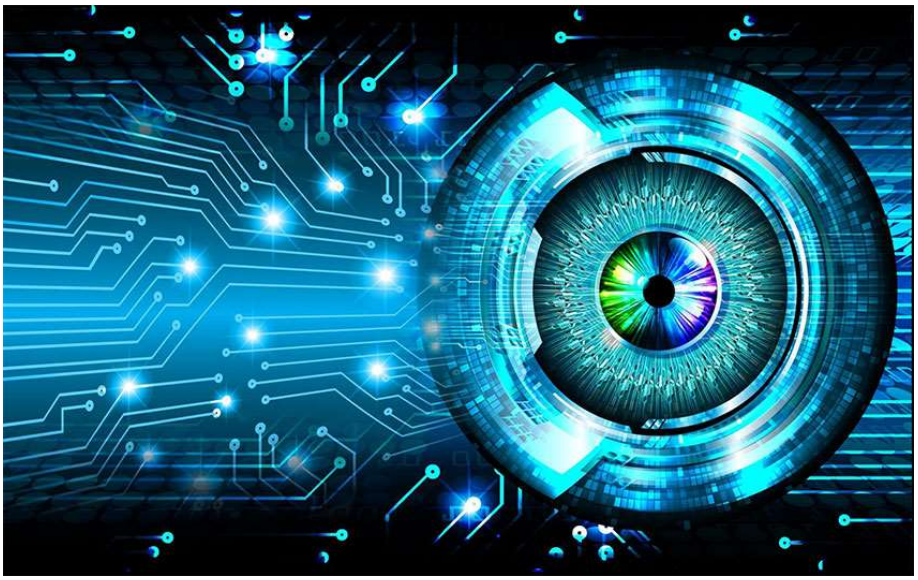
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Προπτυχιακό Μάθημα: «Υπολογιστική Όραση»

Τελευταία Σειρά Προγραμματιστικών Ασκήσεων

<u>Όνομα Φοιτητή – Α.Μ.:</u>

Γεώργιος Κρομμύδας – 3260



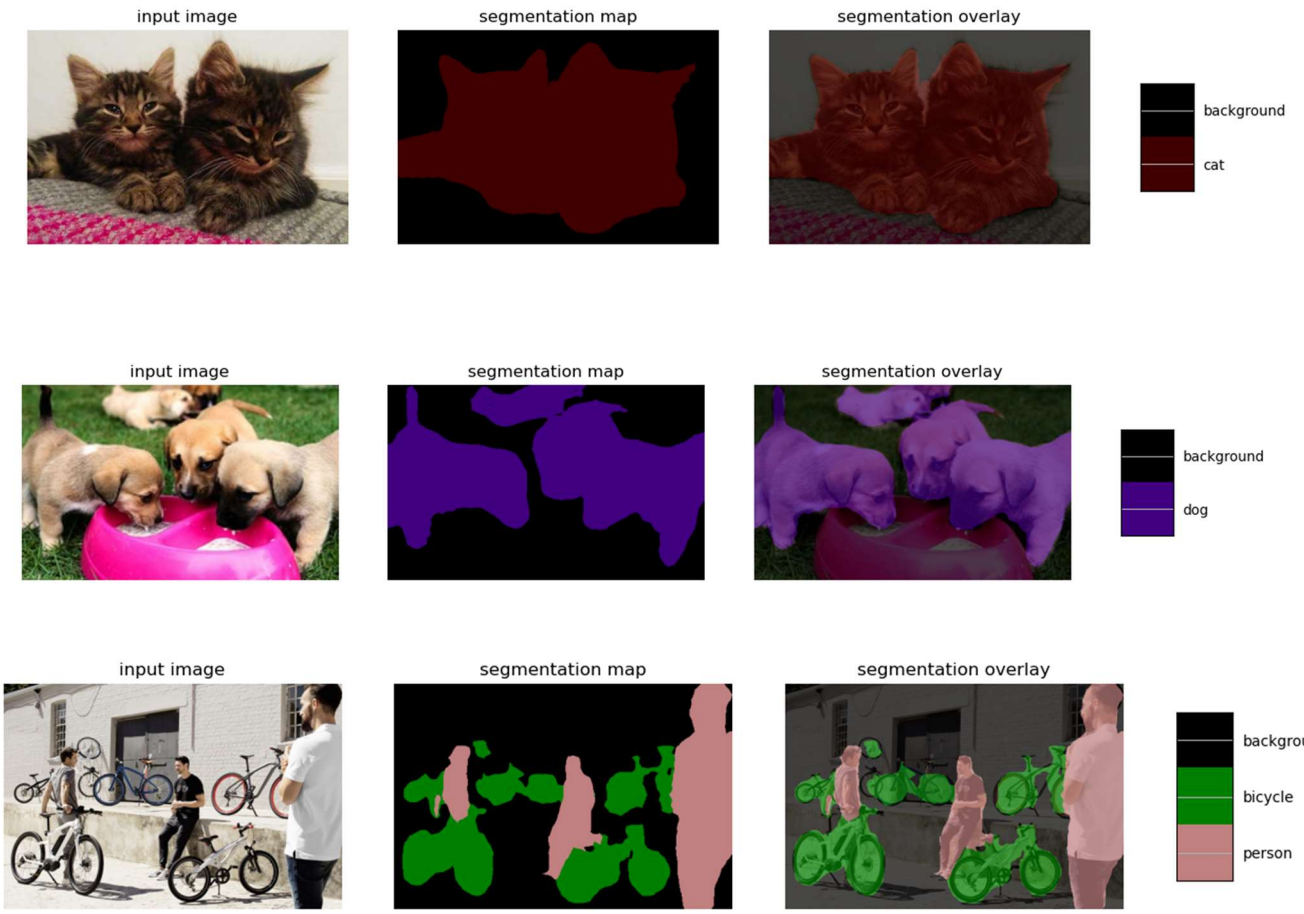
ΙΩΑΝΝΙΝΑ,

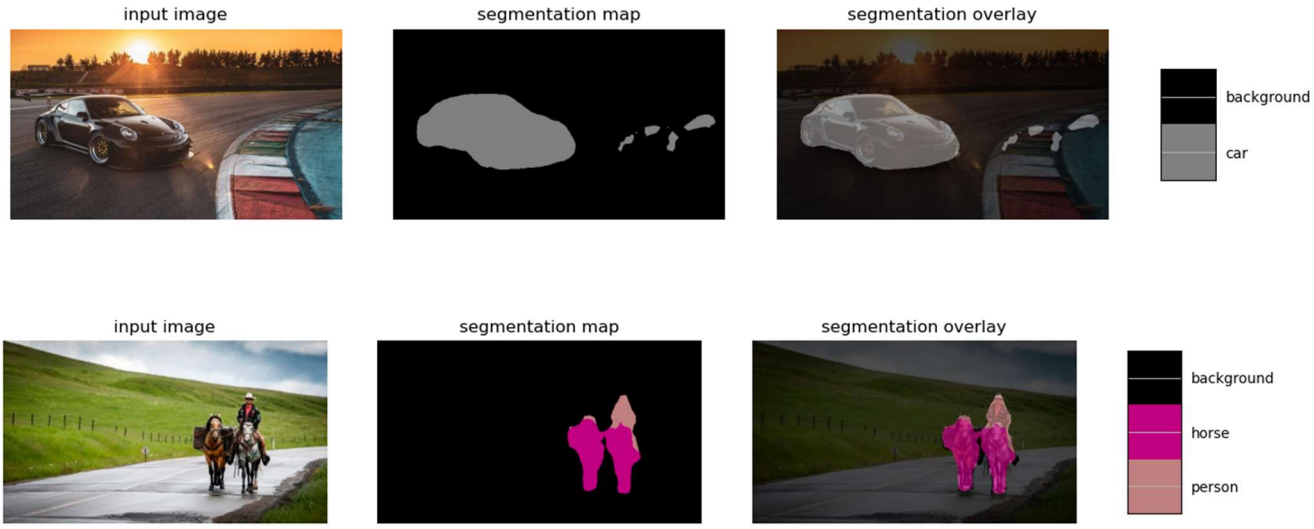
2020

Στην τελευταία εργασία που είχαμε να υλοποιήσουμε, μας δόθηκε ένα συνελλικτικό βαθύ νευρωνικό δίκτυο. Στο πρώτο ερώτημα είχαμε απλά να μετασχηματίσουμε την είσοδο του από το url σε επιλογή εικόνας από τον αντίστοιχο φάκελο του υπολογιστή. Αυτό έγινε απλά με την εντολή **Image.open()**. Έτσι, θα πάρουμε ως είσοδο την αντίστοιχη εικόνα που θέλουμε να κάνουμε σημασιολογική κατάτμηση.

```
204   original_im = Image.open('image2.jpg')
205   resized_im, seg_map = MODEL.run(original_im)
206   vis_segmentation(resized_im, seg_map)
```

Αφού εκτελεστεί το μοντέλο, τότε ως έξοδο θα έχουμε την σημασιολογική κατάτμηση της εικόνας. Δηλαδή θα κατηγοριοποιήσει τα αντικείμενα της εικόνας με το κατάλληλο χρώμα. Κάποια παραδείγματα είναι τα εξής:





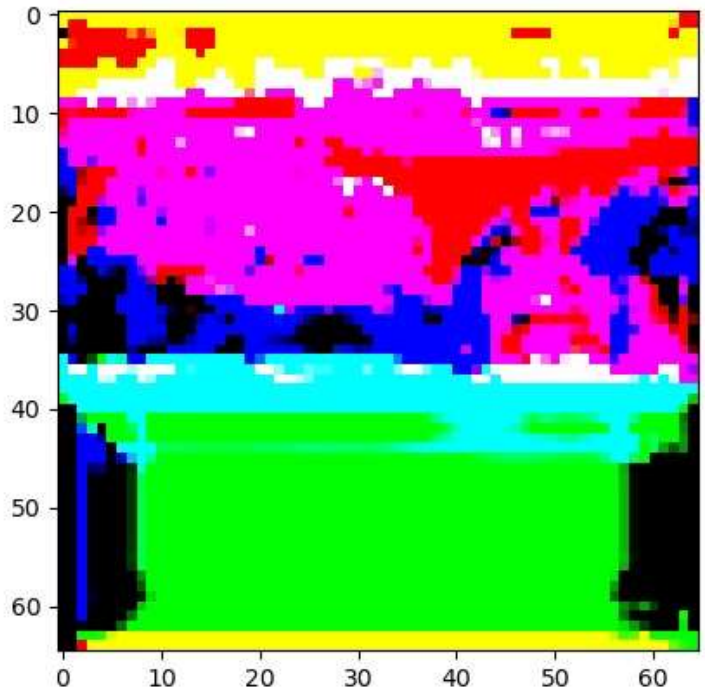
Στο δεύτερο ερώτημα είχαμε να κάνουμε μείωση διάστασης της εικόνας από ενός ενδιάμεσου στρώματος, χρησιμοποιώντας τον αλγόριθμο **PCA(n_components=3)**. Το ζητούμενο ήταν να μειώσουμε την εικόνα σε διάσταση **d=3** και να το οπτικοποιήσουμε. Για να πάρουμε την εικόνα ενός ενδιάμεσου στρώματος χρησιμοποιήθηκε η **MODEL.sess.get_operations()** έτσι ώστε να τυπωθεί ο γράφος για την εύρεση ενός ενδιάμεσου στρώματος. Παρακάτω φαίνεται ο κώδικας που υλοποιεί την παραπάνω διαδικασία:

```
MIDDLE_TENSOR_LAYER = 'MobilenetV2/expanded_conv_9/output:0'
```

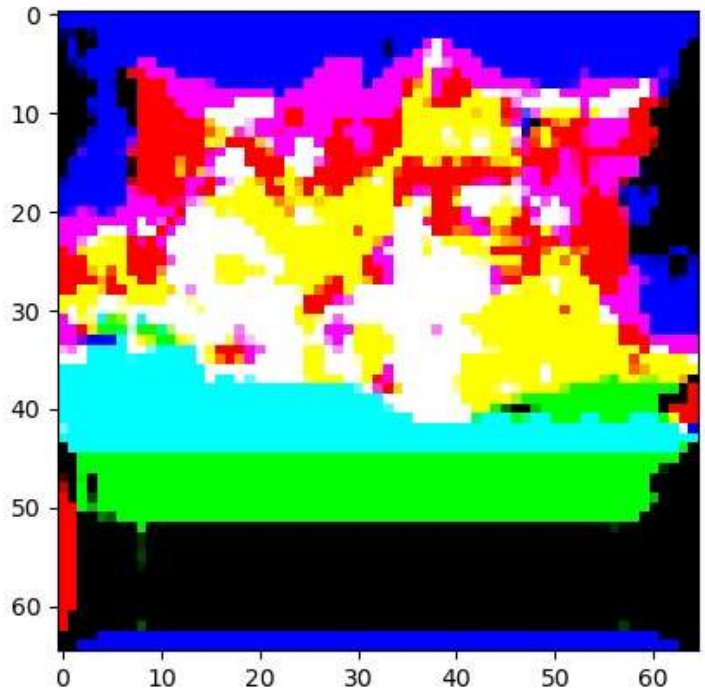
```
    batch_seg_im = self.sess.run(
      self.MIDDLE_TENSOR_LAYER,
      feed_dict={self.INPUT_TENSOR_NAME: [np.asarray(resized_image)]})
    segment_im = batch_seg_im[0]
    return resized_image, seg_map, segment_im
```

```
############ PCA with 3 dimensions ############
seg_im = np.asarray(segment_im)
N = seg_im.shape[0]*seg_im.shape[1]
C = seg_im.shape[-1]
X = np.reshape(seg_im, [N, C])
print('Τα αρχικά δεδομένα μου έχουν μέγεθος: {}'.format(X.shape))
Xreduced = PCA(n_components=3).fit_transform(X)
print('Μετά το PCA έχουμε μέγεθος: {}'.format(Xreduced.shape))
seg_im_reduced = np.reshape(Xreduced, [seg_im.shape[0], seg_im.shape[1], 3])
print(seg_im_reduced.shape)
plt.imshow(seg_im_reduced)
plt.show()
```

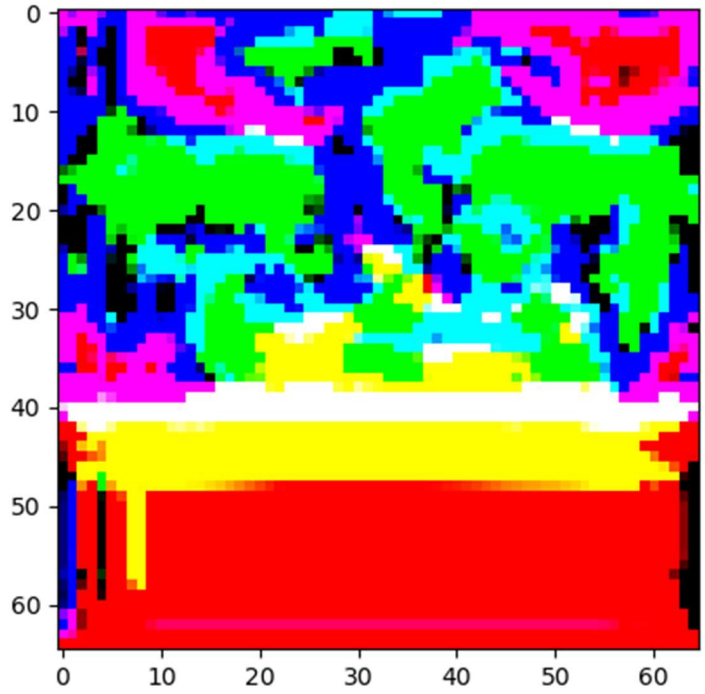
Κατά την εκτέλεση του παραπάνω block, τότε οι διάσταση των δεδομένων του κρυφού επιπέδου θα γίνει 3, όσο και η διάσταση που χρειαζόμαστε έτσι ώστε να τα οπτικοποιήσουμε. Παρακάτω φαίνονται κάποια παραδείγματα:



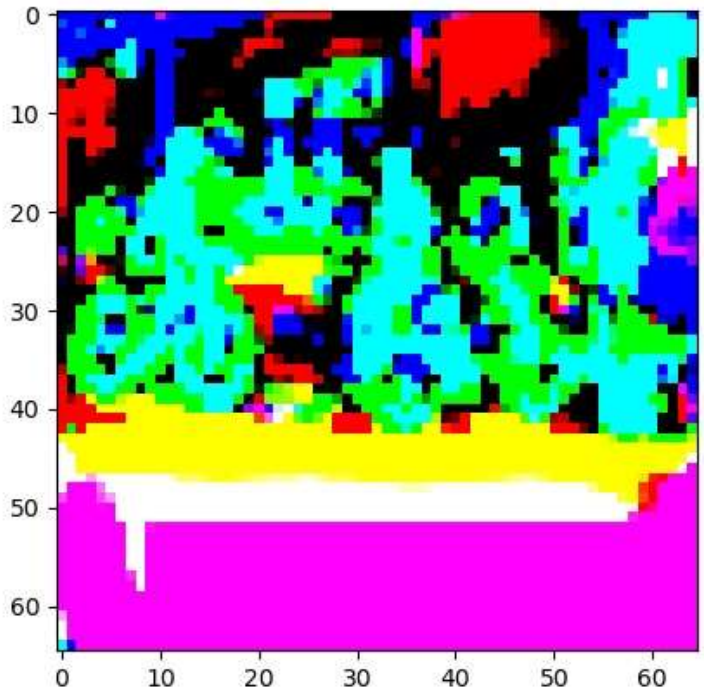
Εικόνα 1 car



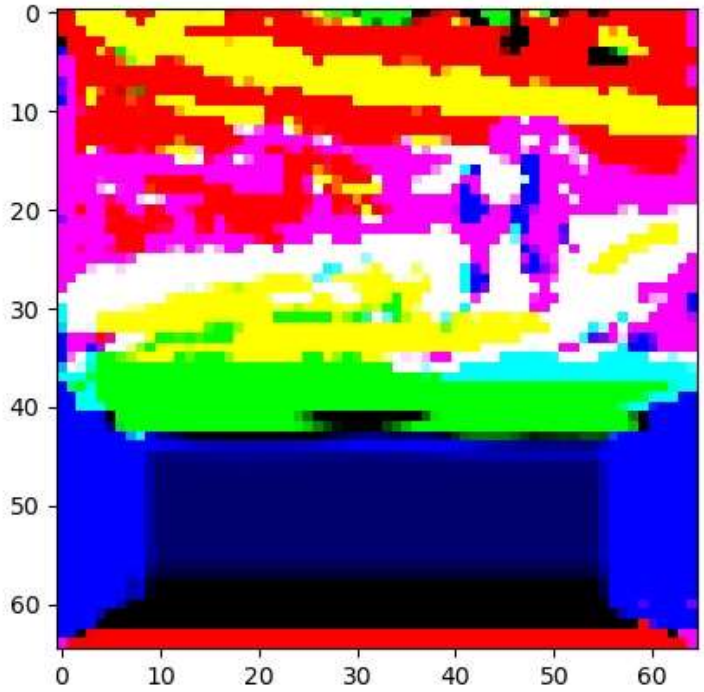
Εικόνα 2 small 1



Εικόνα 3 puppies 1



Εικόνα 4 bmw 1



Εικόνα 5 horse 1

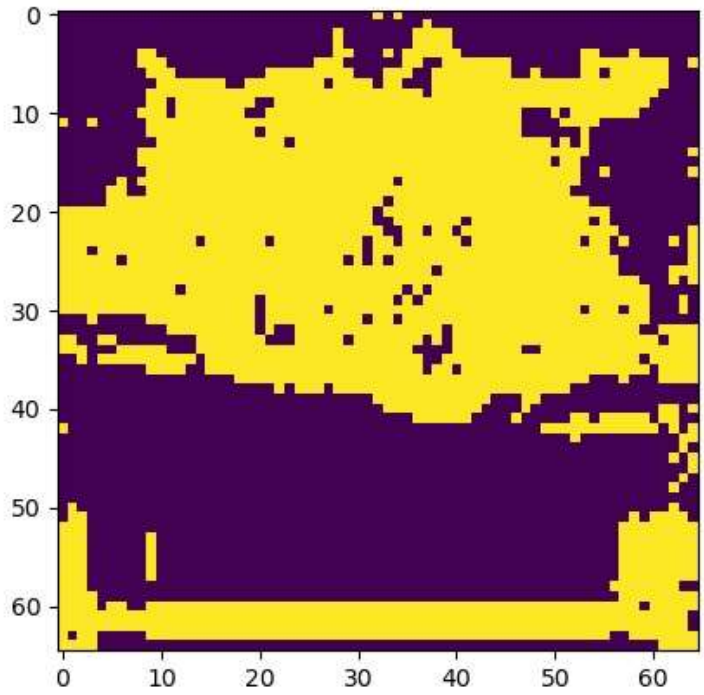
Στο τρίτο και τελευταίο ερώτημα είχαμε πάλι να χρησιμοποιήσουμε τον αλγόριθμο PCA, όμως με την διάσταση των δεδομένων να είναι στα 8. Δηλαδή χρησιμοποιούμε την συνάρτηση **PCA(n_components=8)** για να μειωθεί η διάσταση από d του κρυφού επιπέδου σε **d=8**. Στη συνέχεια εφαρμόζουμε τον αλγόριθμο Kmeans έτσι ώστε να κατατμήσουμε την εικόνα σε δύο ομάδες. Ο αλγόριθμος Kmeans μας βοηθάει ως προς την οπτικοποίηση των αποτελεσμάτων, καθώς δεν είναι εφικτό να υπάρξει οπτικοποιήση των 8 διαστάσεων. Επιπλέον, ο μηχανισμός αναγνώρισης κατηγοριοποιεί τα αντικείμενα της εικόνας βάσει του κέντρου που βρίσκονται. Παρακάτω φαίνεται ο κώδικας:

```
############ PCA with 8 dimensions and KMeans ############
seg_im = np.array(segment_im, dtype = np.uint8)
N = seg_im.shape[0]*seg_im.shape[1]
C = seg_im.shape[-1]
X2 = np.reshape(seg_im, [N, C])
print('Τα αρχικά δεδομένα μου έχουν μέγεθος: {}'.format(X2.shape))
Xreduced2 = PCA(n_components=8).fit_transform(X2)
print('Μετά το PCA έχουμε μέγεθος: {}'.format(Xreduced2.shape))
seg_im_reduced2 = np.reshape(Xreduced2, [seg_im.shape[0], seg_im.shape[1], 8])
```

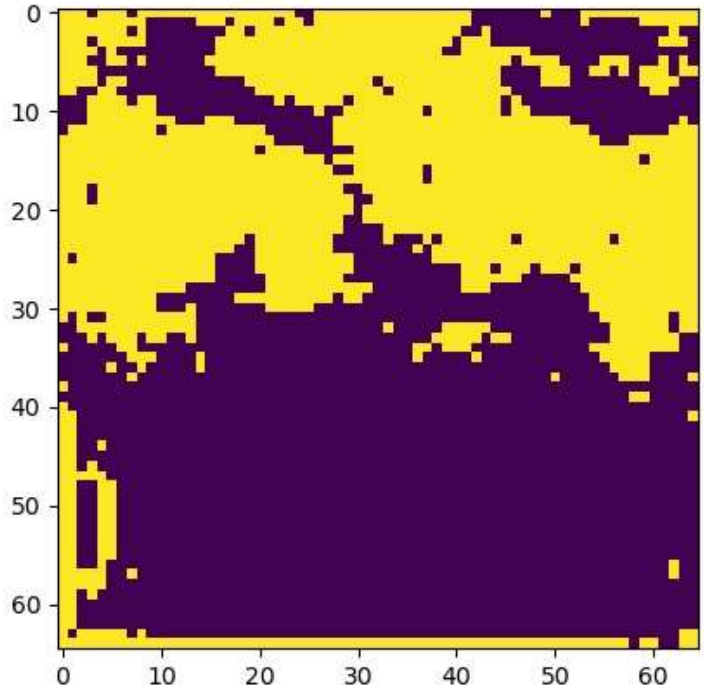
```
# KMeans Algorithm #

Seg_im = np.array(seg_im_reduced2)
Im = np.mean(Seg_im, axis = -1)
X = Im.flatten()
K = 2 # Number of clusters
N = len(X)  # Number of pixels
iterations = 30 # Μέγιστος αριθμός επαναλήψεων του αλγόριθμου
# New centroids
centroids = np.random.rand(K) * 255
# Εκτύπωση αρχικών κέντρων
for k in range(K):
    print('Αρχική τιμή για κέντρο {}: {}'.format(k, centroids[k]))
    for i in range(iterations):
      print('Επανάληψη αρ.{}'.format(i+1))
      distances_from_centroids = np.zeros([K, N])
      for k in range(K):
        distances_from_centroids[k,:] = (X - centroids[k]*np.ones(N,))**2
        # Αυτό το διάνυσμα θα κρατάει τον αριθμό του κοντινότερου κέντρου για το κάθε σημείο
      nearest_centroids = np.argmin(distances_from_centroids, axis=0)
      # Βήμα υπολογισμού νέων κέντρων
      for k in range(K):
          value_seg = X[nearest_centroids == k]
          if(len(value_seg) == 0):
              continue
          centroids[k] = np.mean(value_seg)
print(nearest_centroids.shape)
segmentation = np.reshape(nearest_centroids, [65, 65])
print(segmentation.shape)
plt.imshow(segmentation)
plt.show()
```

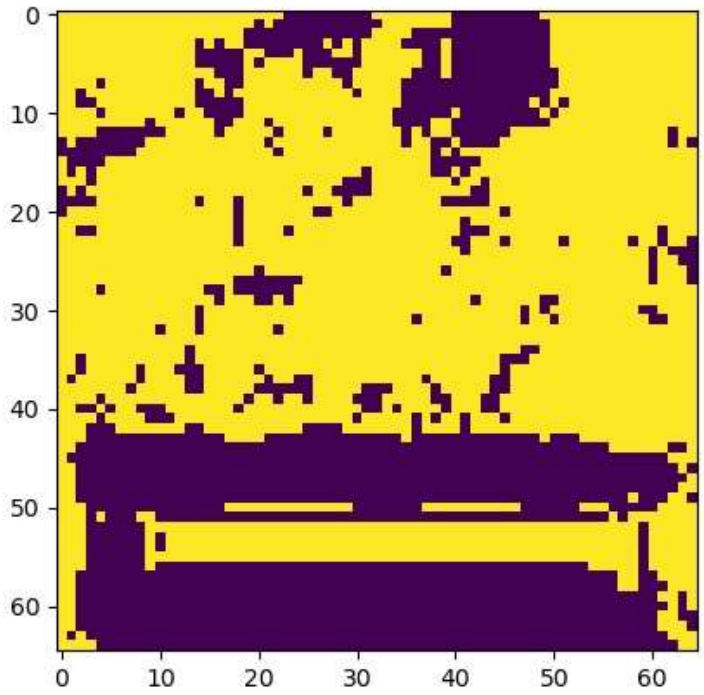
Παρακάτω φαίνονται κάποια παραδείγματα:



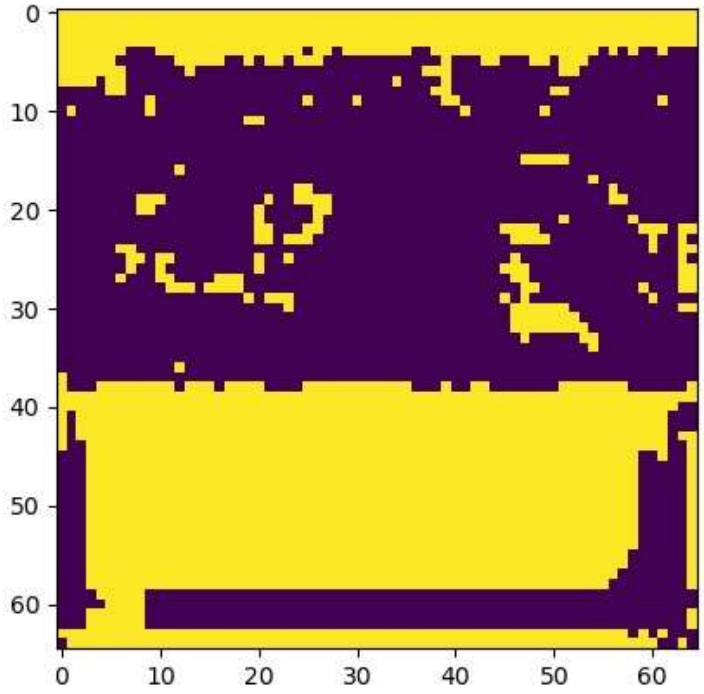
Εικόνα 6 small 2



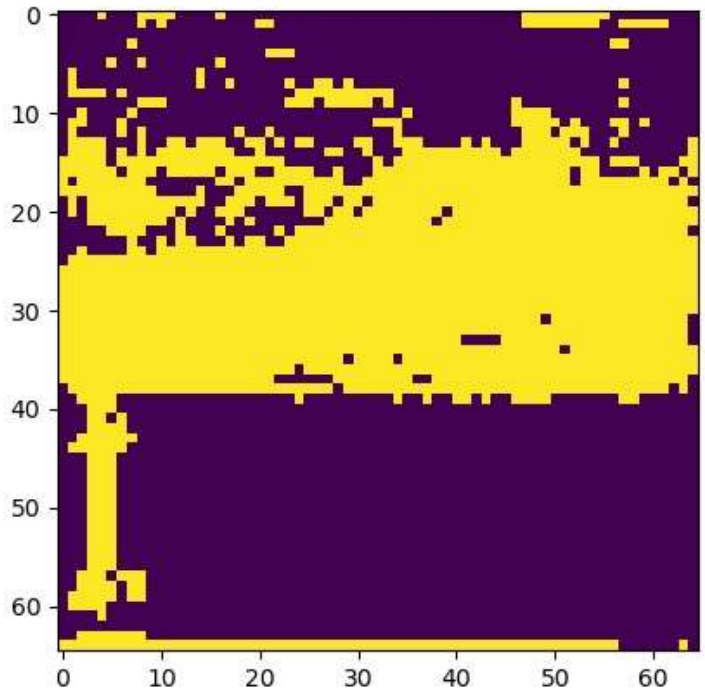
Εικόνα 7 puppies 2



Εικόνα 8 bmw 2



Εικόνα 9 car 2



Εικόνα 10 horse 2

Παρατηρούμε, πως η εφαρμογή των παραπάνω αλγορίθμων δεν βγάζουν πάντα σωστά αποτελέσματα. Για παράδειγμα ο αλγόριθμος PCA τριών διαστάσεων δεν κατάφερε να αναγνωρίσει το αυτοκίνητο ούτε τα άλογα. Ωστόσο, αναγνώρισε τα κουτάβια και τις γάτες. Αντίστοιχα, ο αλγόριθμος PCA οχτώ διαστάσεων με χρήση του αλγορίθμου KMeans για κατάτμηση, είχε επίσης θέμα αναγνώρισης του αυτοκινήτου και των αλόγων.