COMPUTER VISION LAB EXERCISE 1

ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΑΡΑΪΣΚΟΣ
ΑΜ: 1072636

February 27, 2024

Δημιουργία Γκαουσιανής Πυραμίδας

Ερώτημα 3

Για να διαπιστώσουμε ότι η κρουστική απόκριση της σχέσης [5] αποτελεί διακριτό ισοδύναμο του Gaussian πυρήνα, μπορούμε να συγκρίνουμε τις τιμές που παίρνει η διακριτή προσέγγιση του Gaussian πυρήνα για συγκεκριμένο σ. Συγκεκριμένα, παίρνουμε τις τιμές του Gaussian πυρήνα για τις διακριτές τιμές, από -2 έως 2, τις κανονικοποιούμε και αφαιρούμε από αυτές τις κανονικοποιημένες τιμές της h. Όσο πιο κοντά στο μηδέν είναι η διαφορά τους, τόσο πιο καλή η προσέγγιση του πυρήνα από την h. Για σ=1.1, παίρνουμε μία πολύ καλή προσέγγιση του πυρήνα από τη σχέση [5]. Παρακάτω δίνεται ο κώδικάς για την επαλήθευσή του παραπάνω, μαζί με την έξοδό του:

```
sigma = 1.1;
n = -2:2;  % range of values for n for the Gaussian kernel

% Continuous Gaussian kernel
g_continuous = exp(-n.^2 / (2 * sigma^2)) / (sqrt(2 * pi) * sigma);

% Normalize the kernel
g_continuous = g_continuous / sum(g_continuous);

% Impulse response
h = [1 4 6 4 1] / 16;

% Normalize the impulse response
h = h / sum(h);

% Compare the normalized kernels
difference_continuous = g_continuous - h;

disp('Continuous Gaussian Kernel:');
disp(g_continuous);

disp('Normalized Impulse Response (h):');
disp(h);

disp('Difference between Continuous Gaussian Kernel and h:');
disp(difference_continuous);
```

```
Output:
Continuous Gaussian Kernel:
  Columns 1 through 5
    0.0708    0.2445    0.3695    0.2445    0.0708

Normalized Impulse Response (h):
  Columns 1 through 5
    0.0625    0.2500    0.3750    0.2500    0.0625


Difference between Continuous Gaussian Kernel and h:
  Columns 1 through 5
    0.0083   -0.0055   -0.0055   -0.0055    0.0083
```

Ερώτημα 5

Παρακάτω δίνεται ο κώδικας για τη δημιουργία Laplacian πυραμίδας από μια Gaussian και αντίστροφο:

```
%% Create gaussian pyramid from the laplacian and vice versa
% We consider that the top level of both the Gaussian and Laplacian
% pyramid consists of the same image.
% We can create a Laplacian pyramid by substracting from each level
% of the Gaussian pyramid the expanded version of the level above that one.
% By adding the expanded version of the level above of each level, in the
% Gaussian pyramid, to the corrisponding level of the Laplacian pyramid
% we can construct the Gaussian pyramid.
% Gi = Li + expand(Gi+1) / Li = Gi - expand(Gi+1)


img=imread('apple.jpg');

level=4;
lPyr=cell(1,level);
gPyr=cell(1,level);
pyr=cell(1,level);
pyr{1}=im2double(img);

% First create the Gaussian pyramid and then from that the Laplacian
for p=2:level
        pyr{p}=pyr_reduce(pyr{p-1});
end

% adjust the image size
for p=level-1:-1:1
        osz=size(pyr{p+1})*2-1;
        pyr{p}=pyr{p}(1:osz(1),1:osz(2),:);
end

% Finally create the Laplacian pyramid
for p=1:level-1
        lPyr{p}=pyr{p}-pyr_expand(pyr{p+1});
end

lPyr{level} = pyr{level};

% Re-create the Gaussian pyramid from the Laplacian
gPyr{level}=lPyr{level};
```

```
for p=level-1:-1:1
        gPyr{p}=lPyr{p}+pyr_expand(gPyr{p+1});
end

figure(1);
for p=level:-1:1
        subplot(level,1,p);imshow(gPyr{p});
end

figure(2);
for p=level:-1:1
        subplot(level,1,p);imshow(lPyr{p});
end
```

Ερώτημα 6

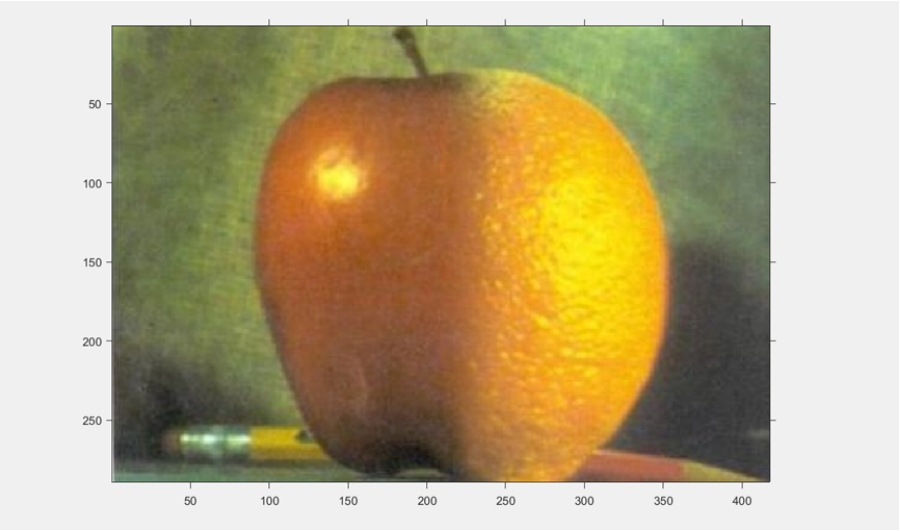
- *pyr_Expand*(): Μέσω αυτής της συνάρτησης κάνουμε up-sample το επίπεδο n+1 της Gaussian πυραμίδας, προκειμένου να το χρησιμοποιήσουμε για τη δημιουργία του επιπέδου n της Laplacian πυραμίδας. Δημιουργούμε αρχικά έναν 1D πυρήνα που καθορίζει το βάρος με το οποίο θα συνεισφέρει κάθε pixel της εικόνας εισόδου, στις interpolated τιμές της διευρυμένης εικόνας. Έπειτα, από τον 1D πυρήνα, δημιουργούμε τον 2D πυρήνα που θα χρησιμοποιήσουμε για να κάνουμε up-sample την εικόνα εισόδου. Κανονικοποιούμε τον πυρήνα πολλαπλασιάζοντας x4 ώστε να διατηρήσουμε την ενέργεια κατά το interpolation. Για να συνθέσουμε τη διευρυμένη εικόνα, κάνουμε διαπέραση σε κάθε μία από τις διαστάσεις της εικόνας εισόδου. Για κάθε επίπεδο, δημιουργούμε μία padded εικόνα ως προς τις γραμμές και μία ως προς τις στήλες και έπειτα εφαρμόζουμε τα διαφορετικά μέρη του πυρήνα για να δημιουργήσουμε τα κομμάτια που θα απαρτίζουν την τελική διευρυμένη εικόνα. Τέλος, ενώνουμε τα κομμάτια για να δημιουργήσουμε το εκάστοτε διευρυμένο επίπεδο της εικόνας.
- *pyr_Reduce*(): Μέσω αυτής της συνάρτησης κάνουμε down-sample την εικόνα εισόδου. Τη χρησιμοποιούμε για να δημιουργήσουμε την Gaussian πυραμίδα. Δημιουργούμε και πάλι έναν 2D πυρήνα, όπως και στην *pyr_Expand*(), χωρίς όμως να απαιτείται εδώ η κανονικοποίησή του. Για να συνθέσουμε την down-sampled εικόνα, ενεργούμε πάνω σε κάθε διάσταση της εικόνα εισόδου. Συγκεκριμένα, εφαρμόζουμε τον Gaussian πυρήνα πάνω σε κάθε διάσταση και συνθέτουμε κάθε "μειωμένο" επίπεδο, επιλέγοντας ανά δύο κάθε pixel της φιλτραρισμένης εικόνας.
- *pyr_Reconstruct*(): Με αυτή τη συνάρτηση συνθέτουμε από μια Laplacian πυραμίδα την εικόνα. Συγκεκριμένα, ενισχύουμε κάθε επίπεδο με τις λεπτομέρειες του επιπέδου πάνω από αυτό και καταλήγουμε στο πρώτο επίπεδο, το οποίο περιέχει την τελική ενισχυμένη εικόνα.
- *gen_Pyr*(): Με αυτή τη συνάρτηση δημιουργούμε Gaussian και Laplacian πυραμίδες. Οι Gaussian πυραμίδες δημιουργούνται μέσω της *pyr_Reduce*(), που αναφέραμε παραπάνω, και οι Laplacian μέσω της *pyr_Expand*() και με χρήση της Gaussian πυραμίδας που δημιουργήθηκε για την εικόνα.
- *pyr_Blend.m* Αποτελεί τον κώδικα με τον οποίο συνθέτουμε μία ομοιόμορφα ενωμένη εικόνα ενός πορτοκαλιού και ενός μήλου. Αρχικά για τις δύο εικόνες, αφού της φέρουμε στο ίδιο μέγεθος, δημιουργούμε μια Laplacian πυραμίδα, 5 επιπέδων, για κάθε μία εξ αυτών. Έπειτα, δημιουργούμε μία μάσκα για κάθε μία, οι οποίες ορίζουν το εύρος της κάθε εικόνας που θα χρησιμοποιηθεί για την συνένωσή τους. Για να είναι ομοιόμορφα ενωμένες οι δύο εικόνες, εφαρμόζουμε στις μάσκες ένα Gaussian φίλτρο, ώστε να υπάρχει ομοιόμορφη μετάβαση από την περιοχή των που θα χρησιμοποιήσουμε προς αυτή που δεν θα χρησιμοποιήσουμε σε κάθε εικόνα. Για να δημιουργήσουμε τη Laplacian πυραμίδα της ενωμένης εικόνας, εφαρμόζουμε την αντίστοιχη μάσκα σε κάθε επίπεδο της Laplacian πυραμίδας της κάθε εικόνας και ενώνουμε αυτές τις εικόνες για να δημιουργήσουμε το αντίστοιχο επίπεδο της πυραμίδας της ενωμένης εικόνας. Τέλος, για να ανακτήσουμε την ενωμένη εικόνα, χρησιμοποιούμε την *pyr_Reconstruct*() στην Laplacian πυραμίδα που δημιουργήσαμε. Εναλλακτικά, επίσης μπορούμε να εφαρμόσουμε κατευθείαν τις μάσκες πάνω στις αρχικές εικόνες και να τις ενώσουμε. Στη συγκεκριμένη περίπτωση το οπτικό αποτέλεσμα δεν έχει κάποια αισθητή διαφορά.

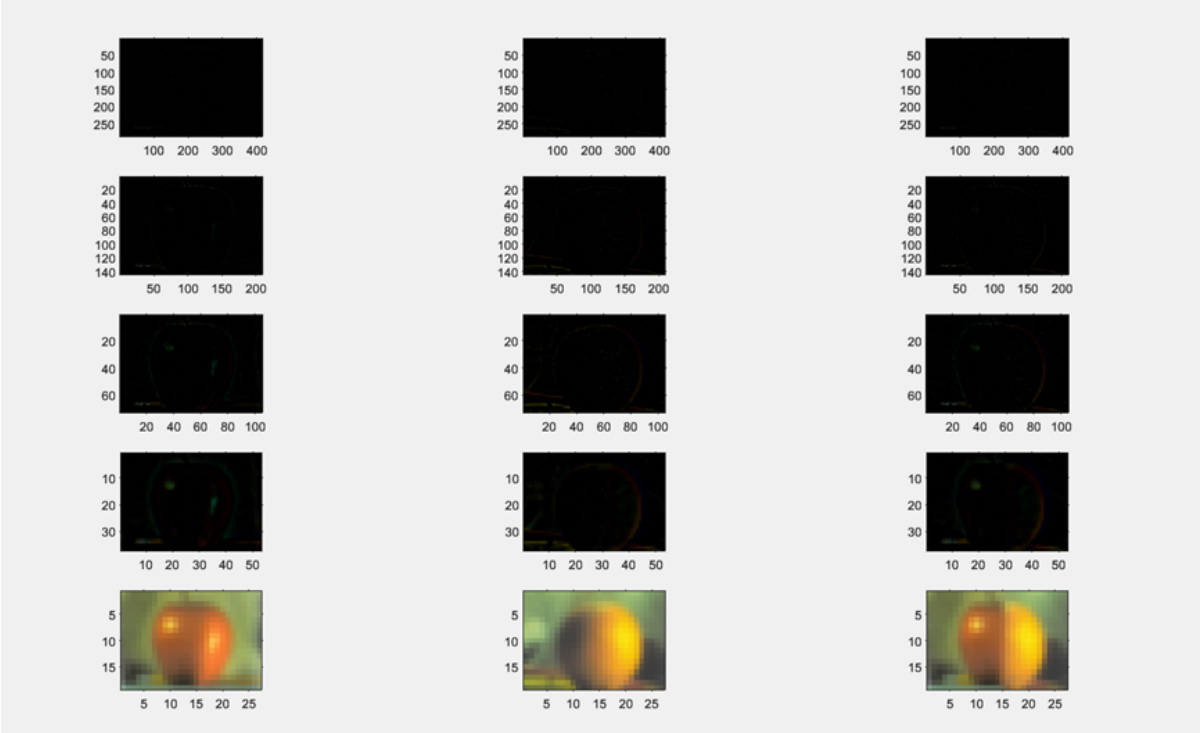
Διαδικασία σελίδα 9

Ερώτηση 1

Παρακάτω δίνεται ο κώδικας για τη δημιουργία των ζητούμενων πυραμίδων:

```
    imga = im2double(imread('apple.jpg'));
imgo = im2double(imread('orange.jpg'));
imga = imresize(imga,[size(imgo,1) size(imgo,2)]);
[M N ~] = size(imga);
v = 230;
level = 5;
% the Laplacian pyramids
limga = genPyr(imga,'lap',level);
limgo = genPyr(imgo,'lap',level);
maska = zeros(size(imga));
maska(:,1:v,:) = 1;
masko = 1-maska;
blurh = fspecial('gauss',30,15);
maska = imfilter(maska,blurh,'replicate');
masko = imfilter(masko,blurh,'replicate');
% the blended pyramid
limgb = cell(1,level);
for p = 1:level
        [Mp Np ~] = size(limga{p});
        maskap = imresize(maska,[Mp Np]);
        maskor= imresize(masko,[Mp Np]);
        limgb{p} = limga{p}.*maskap + limgo{p}.*maskor;
end
imgb = pyrReconstruct(limgb);
figure(1),imshow(imgb) % blend by pyramid

figure(2)
for i=1:level
        subplot(5,3,(i-1)*3+1);
        imshow(limga{i});

        subplot(5,3,(i-1)*3+2);
        imshow(limgo{i});

        subplot(5,3,(i-1)*3+3);
        imshow(limgb{i});
end
```





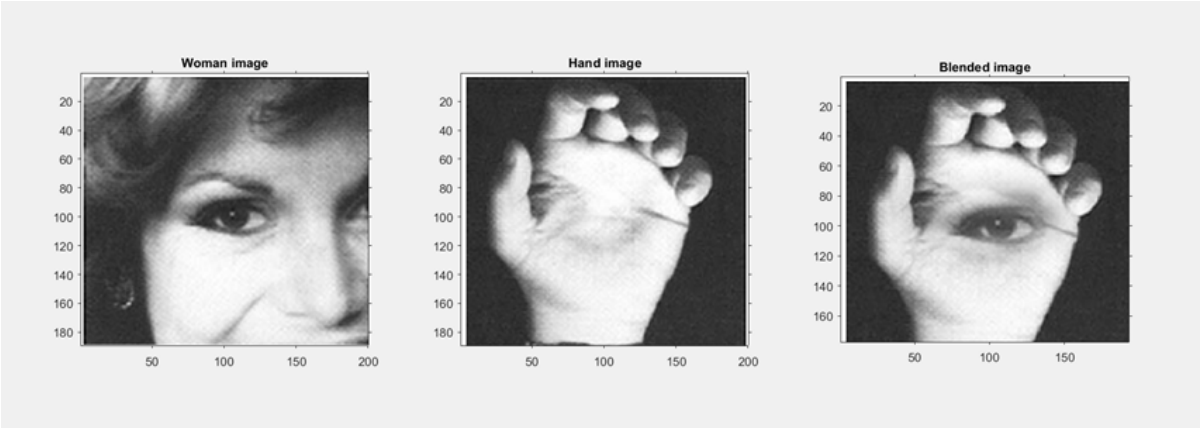
Παραπάνω βλέπουμε τις Laplacian πυραμίδες που σχηματίζονται για τις τρεις εικόνες. Παρατηρούμε ότι σε κάθε πυραμίδα, αρχικά δεν μπορούμε να διακρίνουμε κάποιο χαρακτηριστικό (π.χ. κάποια ακμή), αλλά όσο ανεβαίνουμε επίπεδα φαίνεται το σχήμα από το φρούτο και τέλος η εικόνα του φρούτου θολή. Αυτό συμβαίνει πιθανός λόγω του πυρήνα kronecker που δημιουργήσαμε σε συνδυασμό με την υποδειγματοληψία, επιδρούν στα χρώματα της εικόνας. Συγκεκριμένα, ο πυρήνας εφαρμόζεται σε κάθε κανάλι χρώματος ξεχωριστά, η παράμετρος cw ορίζει τη συνεισφορά του αρχικού pixel στο υποδειγματοληπτισμένο pixel και αφού η τελευταία είναι σχετικά μικρή, οδηγούμαστε λογικά σε απώλεια πληροφορίας.
Η δεξιά εικόνα του σχήματος 2, που αποτελεί την επιθυμητή εικόνα, μοιάζει να είναι όσο αληθινή όσο και οι αρχικές εικόνες που χρησιμοποιήσαμε για να τη δημιουργήσουμε. Αυτό γιατί, η μάσκα που εφαρμόστηκε σε κάθε επίπεδο της πυραμίδας των δύο εικόνων, φιλτραρίστηκε με έναν Gaussian πυρήνα, προκειμένου να δημιουργήσουμε μια ομαλή μετάβαση από τη μία εικόνα στην άλλη. Σε περίπτωση που δεν είχαμε εφαρμόσει αντίστοιχο πυρήνα, η τελική εικόνα θα έμοιαζε με την αριστερή εικόνα του σχήματος 2.

Ερώτημα 2

Η λογική που ακολουθήσαμε για την δημιουργία της εικόνα του σχήματος 11 είναι παρόμοια με αυτή που ακολουθήσαμε για τη δημιουργία της δεξιάς εικόνας του σχήματος 2. Μετατρέψαμε σε gray-scale τις εικόνες, τις φέραμε στο ίδιο μέγεθος και δημιουργήσαμε τις Laplacian πυραμίδες τους. Έπειτα, με χρήση της imshow() και του εργαλείου data tips, βρήκαμε τις συντεταγμένες για μια ορθογώνια γύρο από το μάτι της γυναίκας. Δημιουργήσαμε μία μάσκα για την εικόνα woman.png με άσσους μόνο στο επιθυμητό ορθογώνιο και την συμπληρωματική μάσκα για την εικόνα hand.png. Τέλος, για κάθε επίπεδο των πυραμίδων των εικόνων που είχαμε δημιουργήσει, ενώσαμε τις εικόνες από κάθε επίπεδο αφού εφαρμόσαμε πρώτα την αντίστοιχη μάσκα πάνω στην κάθε μία.

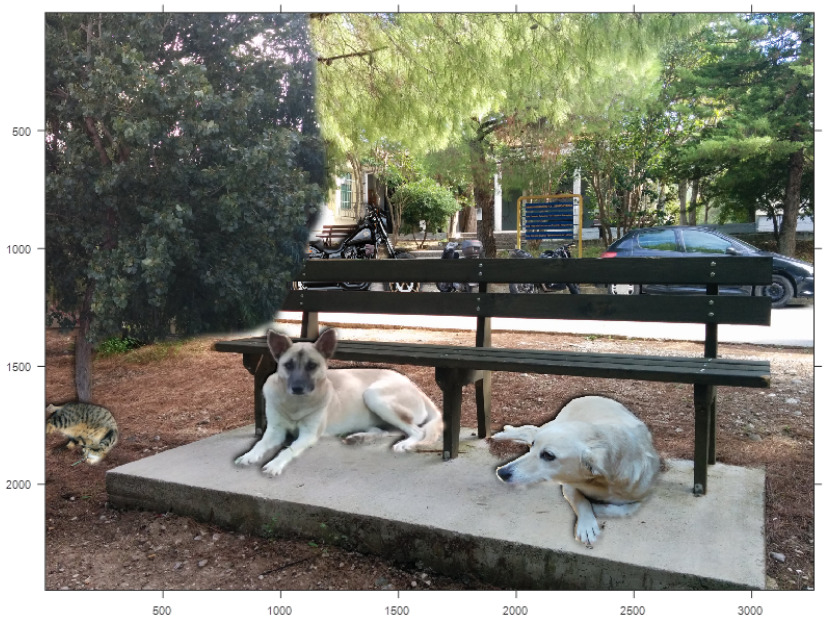
Ο κώδικας για την παραπάνω διαδικασία δίνεται παρακάτω, καθώς και τα αποτελέσματά του:

```
img_woman=rgb2gray(imread('woman.png'));
img_hand=rgb2gray(imread('hand.png'));
img_woman=imresize(img_woman,[size(img_hand,1) size(img_hand,2)]);

level=5;
Limg_woman=genPyr(img_woman,'lap',level);
Limg_hand=genPyr(img_hand,'lap',level);

%% dl[64,118], dr[147,118], ul[64,90], ur[147,90]
maskW=zeros(size(img_woman));
maskW(90:118,64:147)=1;
maskH=1-maskW;
filter_size=[28,83];
blurh=fspecial('gauss',filter_size,10);
maskW=imfilter(maskW,blurh,'replicate');
maskH=imfilter(maskH,blurh,'replicate');

Lblended=cell(1,level); % the blended pyramid
for p=1:level
        [Mp Np ~]=size(Limg_woman{p});
        maskWp=imresize(maskW,[Mp Np]);
        maskHp=imresize(maskH,[Mp Np]);
        Lblended{p}=Limg_woman{p}.*maskWp+Limg_hand{p}.*maskHp;
end

blended=pyrReconstruct(Lblended);

% Display the original and decoded images
figure;
subplot(1, 3, 1);
imshow(img_woman);
title('Woman image');
subplot(1, 3, 2);
imshow(img_hand);
title('Hand image');
subplot(1,3,3);
imshow(blended);
title('Blended image');
```



Ερώτημα 3

Παρακάτω φαίνεται η σύνθεση με τις δοσμένες εικόνες, συν μία της επιλογής μου (Harley). Οι ορισμοί της μάσκας της κάθε εικόνας και των πυραμίδων δίνεται στον κώδικά.



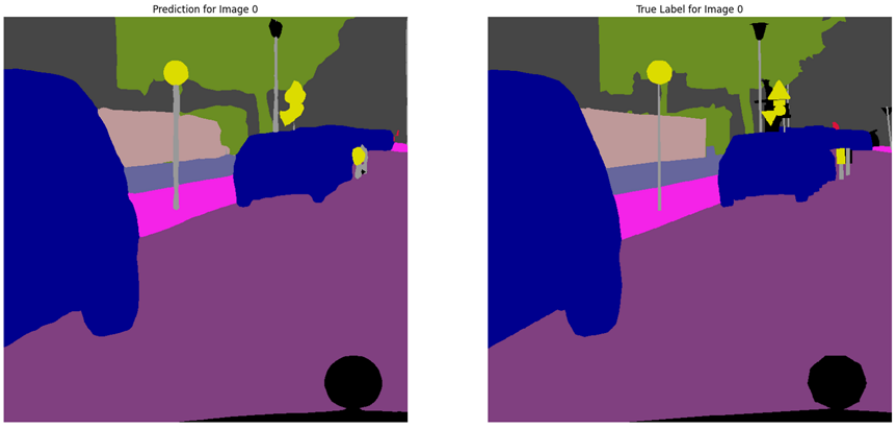
Ερώτημα 4

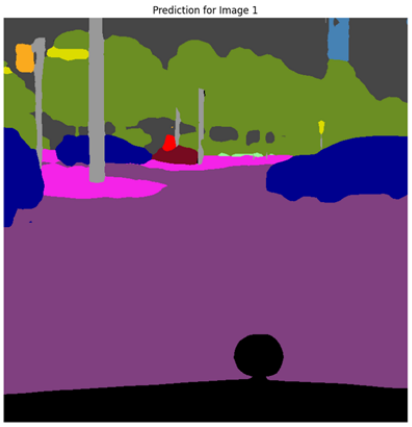
PSPNet

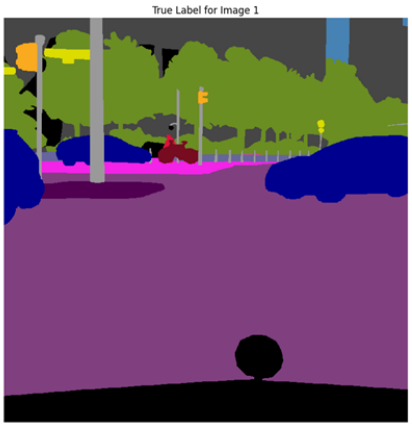
Αφού έχουμε να κάνουμε με αξιολόγηση μοντέλου που πραγματοποιεί κατάτμηση στην εικόνα, θα χρησιμοποιήσουμε ως μετρική αξιολόγησης την Mean IoU. Η Mean IoU είναι ο μέσος όρος των IoUs κάθε κλάσης και μας δείχνει την απόδοση του μοντέλου για όλες τις κλάσεις.
Με βάση τα παρακάτω αποτελέσματα βλέπουμε ότι το μοντέλο έχει αρκετά καλή απόδοση γενικά. Συγκεκριμένα, έχει πολύ καλή απόδοση ως προς τις κλάσεις: 7,8,11,17,19,20,21,22,23,24,26,27,28,32,33, δηλαδή εντοπίζει με αρκετά καλή ακρίβεια αντικείμενα που ανήκουν σε κάποια από τις κλάσεις: road, sidewalk, building, pole, traffic light, traffic sign, vegetation, terrain, sky, person, car, truck, bus, motorcycle, bicycle. Επίσης, μπορούμε να δούμε ότι η διασπορά των κλάσεων είναι αρκετά μικρή, με τη μεγαλύτερη εξ αυτών να μην ξεπερνά το 0.18, πράγμα που ενισχύει την προηγούμενη παρατήρηση για την καλή απόδοση του μοντέλου σε γενικές γραμμές. Τέλος, υπολογίζουμε τη Mean IoU για κάθε εικόνα, ξεχωριστά και έπειτα την

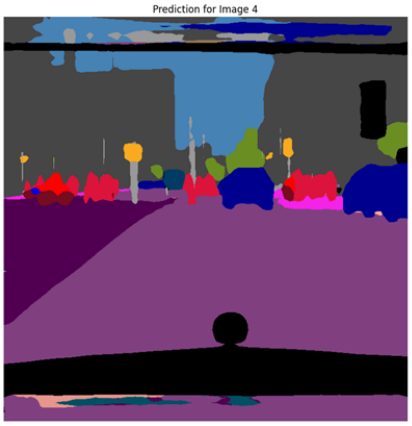
μέση τιμή για όλες τις εικόνες και την διασπορά τους. Παρατηρούμε αρκετά μικρή μέση τιμή εδώ, γεγονός που πιθανώς οφείλεται σε σφάλματα λανθασμένης κατηγοριοποίησης που επηρεάζουν τη μέση τιμή.

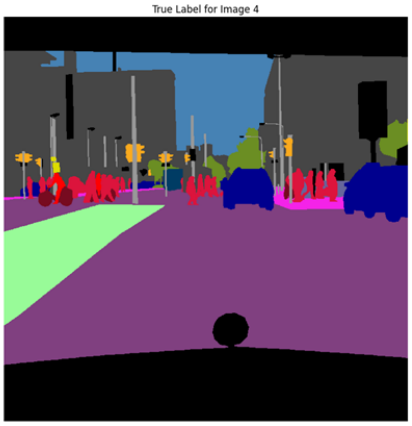
Παρακάτω δίνονται μερικές από τις εικόνες εξόδου του μοντέλου μαζί με το ground truth τους:












```
    Ious per class:
0 :  0.0
1 :  0.9162448208173986
2 :  0.7341297740671573
3 :  0.18497344476042857
4 :  0.24912042685035904
5 :  0.26946246634766313
6 :  0.053747120740244306
7 :  0.9153501434085943
8 :  0.7003714425045603
9 :  0.17141496389347335
10 :  0.003124589202471358
11 :  0.8269556791907802
12 :  0.2610514069266666
13 :  0.4196343732951056
14 :  0.0
15 :  0.24653310696095077
16 :  0.0
17 :  0.5200140995516698
18 :  0.0828675197911642
19 :  0.6202508788217214
20 :  0.654371593726179
21 :  0.8889425071440485
22 :  0.5888205808570374
23 :  0.8546985270873333
24 :  0.7316727952532024
25 :  0.4567740400082855
26 :  0.9054035009026506
27 :  0.6018515207146207
28 :  0.659250666434421
29 :  0.0
30 :  0.006327862368993475
31 :  0.13243910297517197
32 :  0.5131140050262737
33 :  0.6773737061369776
34 :  nan
****************************
Mean IoU: 0.4366554901695766
****************************
Ious variance per class:
0 :  0.0
1 :  0.053913828112216136
```

```
2 :  0.10851731884180064
3 :  0.019291438744258583
4 :  0.04848998389158091
5 :  0.051600495677945855
6 :  0.020960531776140764
7 :  0.069551913763448
8 :  0.11365785248830004
9 :  0.04024081257170692
10 :  2.3403282259863835e-06
11 :  0.08857336776537098
12 :  0.07149806420294025
13 :  0.06637964299090987
14 :  0.0
15 :  0.03728308154547692
16 :  0.0
17 :  0.056616652458409925
18 :  0.008623482477944118
19 :  0.08448850578240975
20 :  0.09553648226987947
21 :  0.06952238533215757
22 :  0.1002264230353269
23 :  0.12266243374515413
24 :  0.1103225306736973
25 :  0.08974749961026245
26 :  0.10563539856946107
27 :  0.09643173002847256
28 :  0.17067952927219765
29 :  0.0
30 :  0.0016863261214808793
31 :  0.027870578657258394
32 :  0.08615003118691829
33 :  0.09901746908008278
34 :  nan
**************************
Ious mean per image:
0 :  0.07247413471377044
1 :  0.11668083433188733
2 :  0.07700092100540724
3 :  0.22286297621756776
4 :  0.23638844067434828
5 :  0.20819478210456444
6 :  0.18126693791296147
7 :  0.24972991853410115
8 :  0.12513159634118942
9 :  0.18006348968720534
10 :  0.20425032336867238
11 :  0.22564285009496407
12 :  0.20130309948619599
13 :  0.09190118988532342
14 :  0.05571923580814619
15 :  0.2665642408178379
16 :  0.15087371956727672
17 :  0.21594391247832054
18 :  0.2200139219826097
19 :  0.1734615559121503
20 :  0.1558844609856432
21 :  0.1721426895556142
```

22 : 0.17303007396770714
23 : 0.2275493870890024
24 : 0.03710764544401435
25 : 0.10838433964719262
26 : 0.047901372031385364
27 : 0.05608668922331802
28 : 0.13939299262229743
29 : 0.20199007045513434
30 : 0.05403518365165457
31 : 0.09443210704783168
32 : 0.21154218221240706
33 : 0.13149202501883497
34 : 0.2218474130332288
35 : 0.17158872067022998
36 : 0.11792347063137366
37 : 0.2550111102483117
38 : 0.2662719248489656
39 : 0.043950540805853054
40 : 0.1284800749016769
41 : 0.14688183954946152
42 : 0.17879867303812338
43 : 0.13123324763189664
44 : 0.2744870947471741
45 : 0.08372998071341595
46 : 0.1641100491737079
47 : 0.19402387695500095
48 : 0.20157719398172433
49 : 0.15377020002095315
50 : 0.12374382940640853
51 : 0.17116922771856907
52 : 0.15191309111322354
53 : 0.24202126453638653
54 : 0.27785512593684086
55 : 0.17215543250078852
56 : 0.12915467366723865
57 : 0.1402526609116807
58 : 0.26485103083468
59 : 0.11710871651172283
60 : 0.24270961132926072
61 : 0.1522506574183707
62 : 0.1579130547888045
63 : 0.2971809341892582
64 : 0.15489803414226938
65 : 0.2726152608754229
66 : 0.19372245808233843
67 : 0.1411409316162242
68 : 0.17209320532929467
69 : 0.12328700330677833
70 : 0.13519129195064078
71 : 0.1742414403812026
72 : 0.14286556611036555
73 : 0.23654477042631905
74 : 0.09503370395109513
75 : 0.1714024675814918
76 : 0.20399515298384086
77 : 0.21321198498013053
78 : 0.12781554597462189

79 : 0.2982361800339586
80 : 0.1978899418816149
81 : 0.18565005246777266
82 : 0.2641543283419341
83 : 0.18211921152239025
84 : 0.2169195282494925
85 : 0.28033200687684806
86 : 0.16524220925709485
87 : 0.04918658129971906
88 : 0.24689679007582851
89 : 0.048498482526997434
90 : 0.23661623310391475
91 : 0.23529918852618228
92 : 0.22839841788405907
93 : 0.09926459757940993
94 : 0.05028154412449941
95 : 0.1829977786966796
96 : 0.056839080804409
97 : 0.20453568395530194
98 : 0.07233682098060518
99 : 0.1147316198736259
100 : 0.15929687506129575
101 : 0.27000198758564486
102 : 0.07993272577011086
103 : 0.18767337540776968
104 : 0.11069181807664319
105 : 0.2381183025597351
106 : 0.25718952818272783
107 : 0.16341379084545363
108 : 0.13772138782878215
109 : 0.1625186034879202
110 : 0.23273543271741443
111 : 0.19449545658282905
112 : 0.21015392014554884
113 : 0.20350941993123964
114 : 0.18180018906164136
115 : 0.18167728817621787
116 : 0.05554843343425052
117 : 0.1968026949860098
118 : 0.10208082134292272
119 : 0.1234680262109675
120 : 0.22859348163414964
121 : 0.05331555314604543
122 : 0.08436092026957892
123 : 0.053727905335321555
124 : 0.2594762825952404
125 : 0.13574282953026426
126 : 0.193226583300211
127 : 0.25227382718316144
128 : 0.26919406157979975
129 : 0.10678267068448424
130 : 0.16314794754661735
131 : 0.1954745240218572
132 : 0.15961471860522516
133 : 0.21885726707874414
134 : 0.22978970621230468
135 : 0.2414237140021843

136 : 0.11103323527273656
137 : 0.1857681929474686
138 : 0.07943248706946565
139 : 0.21049992788908334
140 : 0.1977153308597595
141 : 0.0941141019506702
142 : 0.20876489838800188
143 : 0.08466413122398485
144 : 0.19116756960169173
145 : 0.2644596841751062
146 : 0.22939362968429486
147 : 0.2829150694119442
148 : 0.1054138017432884
149 : 0.227292994049658
150 : 0.10150721296355926
151 : 0.21758053591835086
152 : 0.23906202691554895
153 : 0.25291180250462786
154 : 0.18459357571364982
155 : 0.14033973837657662
156 : 0.19423088411219414
157 : 0.1459145169561632
158 : 0.20288476368930133
159 : 0.21389691449270856
160 : 0.08335344784183706
161 : 0.16455416346797228
162 : 0.17305572309430933
163 : 0.19969714170720385
164 : 0.2278149986582097
165 : 0.1759827145405016
166 : 0.1020691333190988
167 : 0.24310031814341296
168 : 0.32033911800965675
169 : 0.09069492249288985
170 : 0.1452464386829654
171 : 0.22990061808461007
172 : 0.19577093366883241
173 : 0.11675964097385351
174 : 0.12916842439817489
175 : 0.21432947241431088
176 : 0.0940984663922205
177 : 0.1551102395216021
178 : 0.19450745327518507
179 : 0.2029955084298984
180 : 0.15762456023990962
181 : 0.11785010876361467
182 : 0.24116285248395422
183 : 0.22262370722222255
184 : 0.2305212414322624
185 : 0.1757957825661043
186 : 0.18776206494686987
187 : 0.27523654764783456
188 : 0.13440726152586321
189 : 0.18312691269318715
190 : 0.10929500079515125
191 : 0.052458819487015404
192 : 0.25967102421525434

193 : 0.19313577262566003
194 : 0.24940489523642512
195 : 0.19694101006373538
196 : 0.16955814342925532
197 : 0.08094731056928942
198 : 0.2275146243282843
199 : 0.20447920218478097
200 : 0.13695567510465656
201 : 0.10577593033745875
202 : 0.1796331331334965
203 : 0.192351215898147
204 : 0.1035853842035237
205 : 0.1835480951521652
206 : 0.04764076305190246
207 : 0.1500130220864379
208 : 0.24397316739736208
209 : 0.1631320237859705
210 : 0.18813272600594977
211 : 0.20443155491852733
212 : 0.13912743305063432
213 : 0.22945521745901445
214 : 0.13915597831436888
215 : 0.11739372924072253
216 : 0.2539184194902611
217 : 0.08114781708302914
218 : 0.16818476428613732
219 : 0.26026556543273993
220 : 0.18534892018200141
221 : 0.16623840491759298
222 : 0.14609346626430117
223 : 0.16977897655961183
224 : 0.26043903232432297
225 : 0.18419382069158363
226 : 0.23113546132592627
227 : 0.23038128434392635
228 : 0.2172823807894287
229 : 0.21640444898789202
230 : 0.15320144689132814
231 : 0.20306805843188427
232 : 0.12041396841199956
233 : 0.22934103254013644
234 : 0.1597202649318455
235 : 0.11650438944451008
236 : 0.16434524630942665
237 : 0.2144768357180727
238 : 0.2879967644777694
239 : 0.07610594605870641
240 : 0.13907588595722967
241 : 0.20931270370120098
242 : 0.22339341197466647
243 : 0.17881228887070705
244 : 0.09563248848388352
245 : 0.14385235695104573
246 : 0.24530782889641267
247 : 0.17164818234308163
248 : 0.20444325444432268
249 : 0.19838928665059094

250 : 0.12613501885084896
251 : 0.15413142252697642
252 : 0.07640111550068404
253 : 0.23058574254256498
254 : 0.21230682414093383
255 : 0.22797168196593753
256 : 0.15917809517922607
257 : 0.2138724201948666
258 : 0.1951813687623215
259 : 0.1980538614214006
260 : 0.20640224304632823
261 : 0.05523108749092716
262 : 0.13517182990147047
263 : 0.0894462639700037
264 : 0.10967945326206906
265 : 0.1792132351699845
266 : 0.09653575686514189
267 : 0.14079314248595484
268 : 0.24016761525530345
269 : 0.2206550640979796
270 : 0.2223318215343076
271 : 0.21364901862353808
272 : 0.09493638285693369
273 : 0.11181686814984503
274 : 0.21219341478041223
275 : 0.07016057674542725
276 : 0.15184563972262954
277 : 0.15623722233543733
278 : 0.2687288287679815
279 : 0.18520408188504972
280 : 0.027140135859016916
281 : 0.19704728233631588
282 : 0.2355541113578489
283 : 0.133480967568388
284 : 0.09933973284735555
285 : 0.2755625230013781
286 : 0.24658997457037762
287 : 0.19236338035368736
288 : 0.17449845294499938
289 : 0.25251564962135714
290 : 0.19519388576681176
291 : 0.17078480002741123
292 : 0.17253997208056976
293 : 0.1837239106815197
294 : 0.1173940325262346
295 : 0.04052646796266401
296 : 0.19101520768804 27
297 : 0.18929575599631346
298 : 0.25974131053914473
299 : 0.199921162965097
300 : 0.06470545356567306
301 : 0.2619018268928869
302 : 0.12609593686250845
303 : 0.23790742250002522
304 : 0.09188961195397978
305 : 0.14016234725979942
306 : 0.12916531153899008

307 : 0.13993992120888615
308 : 0.1709698842015226
309 : 0.17944230589277577
310 : 0.17450351031791048
311 : 0.09138969116826802
312 : 0.19292769052654946
313 : 0.16030960895014856
314 : 0.12719053564751973
315 : 0.16048335162512817
316 : 0.17187869024385347
317 : 0.19902238139981868
318 : 0.20392603579749818
319 : 0.16891158561377487
320 : 0.20601650866745064
321 : 0.1614330465085472
322 : 0.12829567632912106
323 : 0.24752458695154117
324 : 0.17317637148778683
325 : 0.2523540909588234
326 : 0.1170713278565982
327 : 0.2152483882771848
328 : 0.1354070837932754
329 : 0.171074257519356
330 : 0.21593761825916344
331 : 0.22678661085620927
332 : 0.20971046750357014
333 : 0.19591630538654936
334 : 0.1456767184039293
335 : 0.1684697732380077
336 : 0.10757081332027854
337 : 0.1756868112124636
338 : 0.22358905846159177
339 : 0.20254605028824743
340 : 0.25077829185166706
341 : 0.23195629175323015
342 : 0.1568440963338695
343 : 0.22907860613644904
344 : 0.15839039148760423
345 : 0.263821081531515
346 : 0.16309426458451554
347 : 0.2091802135140557
348 : 0.16536505966031856
349 : 0.11784564211232862
350 : 0.1000290085506282
351 : 0.14110291655213955
352 : 0.20036222060310352
353 : 0.05543951060141199
354 : 0.17463064053751845
355 : 0.1767580308024282
356 : 0.23111660278329849
357 : 0.03534213797017316
358 : 0.19270823278158833
359 : 0.20946054597413802
360 : 0.20848910921507763
361 : 0.19849439051039305
362 : 0.14687711759663
363 : 0.1793250908854394

364 : 0.17454207368324362
365 : 0.09137292626579256
366 : 0.18775473066753529
367 : 0.032771685355113656
368 : 0.06872499195163226
369 : 0.2106630479486286
370 : 0.15356265149239176
371 : 0.159003606281927
372 : 0.15593721566291618
373 : 0.23182218028793783
374 : 0.21048151863957848
375 : 0.2334211685168479
376 : 0.03969967264703537
377 : 0.1912158850527672
378 : 0.2685954714135729
379 : 0.16660038640732364
380 : 0.19197241559235367
381 : 0.06871135668918245
382 : 0.08400607755358833
383 : 0.2826635182424818
384 : 0.15559898909975856
385 : 0.1921975988430278
386 : 0.19417617965192194
387 : 0.08283699867024659
388 : 0.12076986079474337
389 : 0.15441196928983908
390 : 0.29284561137634163
391 : 0.2780565443790964
392 : 0.15980572102932683
393 : 0.22538278827459243
394 : 0.05617548675219116
395 : 0.1977601601537789
396 : 0.23809230851925622
397 : 0.3169943242611934
398 : 0.2603436577015602
399 : 0.25734137089418635
400 : 0.23968855266797953
401 : 0.08298380135910659
402 : 0.2648399767864308
403 : 0.19740641068827108
404 : 0.18923422959598105
405 : 0.1361994351988078
406 : 0.03654410054831286
407 : 0.14081037659872075
408 : 0.13318725347360 03
409 : 0.16511312796054328
410 : 0.051958078598094266
411 : 0.2448839796019864
412 : 0.18143881797026562
413 : 0.12431124654179866
414 : 0.16634911491560161
415 : 0.27792118978663 01
416 : 0.10233265759095025
417 : 0.13692176326159247
418 : 0.1478728249283003
419 : 0.2847277742266803
420 : 0.12680156044443508

421 : 0.18360962809415043
422 : 0.05894423280555677
423 : 0.2207543043500986
424 : 0.15448042718330096
425 : 0.0772267937313714
426 : 0.24200948733420513
427 : 0.06416587959134776
428 : 0.10788700820638462
429 : 0.12455916621634516
430 : 0.20764435709026552
431 : 0.07019846289912274
432 : 0.19836695245158903
433 : 0.2625415660678019
434 : 0.1477992724854949
435 : 0.059288060515981275
436 : 0.21323824913338776
437 : 0.21693056394054708
438 : 0.2342164527846553
439 : 0.03860609752411064
440 : 0.13575018003369568
441 : 0.15630829405244073
442 : 0.27061948687072984
443 : 0.2510340401815238
444 : 0.17211831508224987
445 : 0.15267450179107506
446 : 0.16201905328855806
447 : 0.09165309020123187
448 : 0.06361377388115452
449 : 0.27988701317026904
450 : 0.19982900901750505
451 : 0.13723250396828487
452 : 0.20338437714502894
453 : 0.08380586371664384
454 : 0.1917336700504705
455 : 0.16872249592780594
456 : 0.16527437262384048
457 : 0.2233025564181015
458 : 0.2050555853344299
459 : 0.1923061727479537
460 : 0.14883403094553774
461 : 0.12430095438591753
462 : 0.1333690171090808
463 : 0.18870822478358706
464 : 0.10369703412964491
465 : 0.1870664784356701
466 : 0.19003935511743986
467 : 0.04974316748893707
468 : 0.14167779245574802
469 : 0.21045530062029968
470 : 0.1931838974337092
471 : 0.027775877046010707
472 : 0.09837918043877168
473 : 0.11330851449073198
474 : 0.1596447232202676
475 : 0.14446919429451155
476 : 0.15418180963744785
477 : 0.047270785474087244

```
478 :  0.07112084991916041
479 :  0.16221747766560785
480 :  0.1998788286963931
481 :  0.1917712314864277
482 :  0.11738824613500708
483 :  0.08330826857864244
484 :  0.12027754533097561
485 :  0.1388419467359283
486 :  0.09750598881582308
487 :  0.24960833502984872
488 :  0.2861688272560688
489 :  0.14836969366209177
490 :  0.2507082209946405
491 :  0.15131877996809948
492 :  0.1387414483491892
493 :  0.21709513225821078
494 :  0.25781204412649766
495 :  0.08569691589476543
496 :  0.049411916524184384
497 :  0.02857142857142857
498 :  0.19925581670864417
499 :  0.2598649337513253
****************************
Iou images variance:  0.0039036464948003567
****************************
mean iou images:  0.1703146446370451
****************************
```