Advanced Computer Vision
Programming Exercise 2020

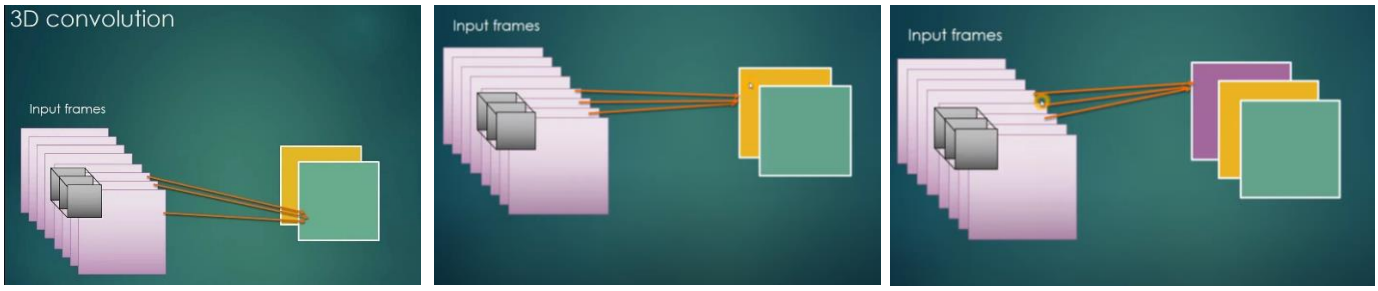
3D Convolution Report

George Kalitsios
Student Number : 62

Η εργασία έχει ως στόχο την κατανόηση της συνέλιξης και την υλοποίησης μιας τρισδιάστατης εκδοχής αυτής σε δεδομένα βίντεο. Το βίντεο αποτελείται από διαδοχικά καρέ διατεταγμένα κατά τον χρονικό άξονα t όπως φαίνεται στην παρακάτω εικόνα, τα καρέ αυτά όλα μαζί μας δίνουν το βίντεο όπως έχουμε διδαχτεί .

Η βασική διαφορά για την εφαρμογή της 3D συνέλιξης σε ένα βίντεο με την δισδιάστατη συνέλιξη 2D σε καρέ είναι ότι στην περίπτωση της τρισδιάστατής συνέλιξης σε βίντεο εφαρμόζουμε την συνέλιξη σε 3 καρέ ταυτόχρονα ώστε να λάβουμε υπόψιν και τις μεταβολές σε διαδοχικά καρέ κατά τον άξονα του χρόνου σε αντίθεση με την δισδιάστατη συνέλιξη που την εφαρμόζουμε ανά 1 καρέ. Έτσι για να πάρω το πρώτο καρέ μετά την εφαρμογή της 3D συνέλιξης θα χρησιμοποιήσω τα 3 πρώτα καρέ (frame 1, 2, 3). Στην συνέχεια για να πάρω το δεύτερο καρέ θα χρησιμοποιήσω τα (frame 2, 3, 4), για το τρίτο καρέ θα χρησιμοποιήσω τα (frame 4, 5, 6) κλπ. όπως φαίνεται στην

παρακάτω εικόνα. Καταλήγω δηλαδή από 150 καρέ να έχω στην έξοδο 148 με την παραπάνω διαδικασία .



Input Αλγορίθμου :

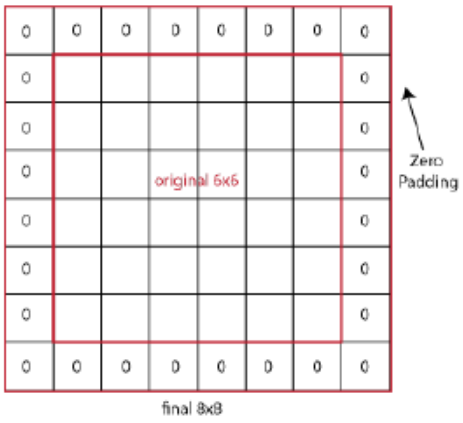
Ο αλγόριθμος δέχεται σαν είσοδο ένα βίντεο με 150 καρέ διάρκειας 5 δευτερολέπτων . Στην παρακάτω εικόνα φαίνονται ενδεικτικά 2 καρέ από το βίντεο .

Output Αλγορίθμου :

Ο αλγόριθμος παράγει ένα βίντεο διάρκειας 5 δευτερολέπτων ελαφρώς θολωμένο κατά τους χωρικούς άξονες και τον άξονα του χρόνου. Στην παρακάτω εικόνα φαίνονται ενδεικτικά 2 καρέ από το βίντεο που προκύπτει από την 3D συνέλιξη .

Συναρτήσεις που υλοποιήθηκαν :

- **pad_image(A, size) :**
  Η συνάρτηση αυτήν χρησιμοποιείται για να κάνουμε zero-padding . Το zero padding είναι χρήσιμο όταν δεν θέλουμε να χάσουμε πληροφορία από τις άκρες τις εικόνας μας αλλά και όταν θέλουμε να αποφύγουμε την μείωση των χωρικών διατάξεων της εικόνας . Στην παρακάτω εικόνα βλέπουμε ένα παράδειγμα zero padding με pad=1



Ο κώδικας φαίνεται στον παρακάτω πίνακα :

```
def pad_image(A, size):
        '''
        your code for volume padding should be placed here
        '''
        padding=int((size-1)/2)
        A_pad_image = np.pad(A, ((0,0), (padding,padding), (padding,padding)), mode='constant',
constant_values = (0,0))
        return A_pad_image
```

τρέχοντας την συνάρτηση με είσοδο ένα τρισδιάστατο πίνακα 5x5 το αποτέλεσμα είναι ένα τρισδιάστατος πίνακας 7x7 με τις τιμές που φαίνονται στον παρακάτω πίνακα :

A.shape = (1, 5, 5) A= [[[1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.]]]	A_pad.shape = (1, 7, 7) A_pad= [[[0. 0. 0. 0. 0. 0. 0.] [0. 1. 1. 1. 1. 1. 0.] [0. 1. 1. 1. 1. 1. 0.] [0. 1. 1. 1. 1. 1. 0.] [0. 1. 1. 1. 1. 1. 0.] [0. 1. 1. 1. 1. 1. 0.] [0. 0. 0. 0. 0. 0. 0.]]]

- **create_smooth_kernel(size)** :

Η συνάρτηση αυτήν δημιουργεί το φίλτρο το οποίο θα χρησιμοποιήσουμε για την συνέλιξη . Μπορεί να χρησιμοποιηθεί και για να δημιουργήσουμε φίλτρα 5x5 , 7x7 κλπ .

Ο κώδικας φαίνεται στον παρακάτω πίνακα :

```
def create_smooth_kernel(size):
        '''
        your code for kernel creation should be placed here
        '''
        return np.ones((size,size,size))/size**3
```

τρέχοντας την συνάρτηση για size=3 το αποτέλεσμα είναι ένα τρισδιάστατο φίλτρο με τις τιμές που φαίνονται στον παρακάτω πίνακα :

```
kernel.shape = (3, 3, 3)
kernel = [[[0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]]
 [[0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]]
 [[0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]
  [0.03703704 0.03703704 0.03703704]]]
```

- **conv_single_step(slice, filt)**:

Αυτήν είναι μια εξτρά απλή συνάρτηση η οποία υλοποιήθηκε και καλείται από την myConv3D για να κάνει 1 απλό βήμα της συνέλιξης δηλαδή αν έχουμε ένα 3x3 kernel και πάρουμε ένα τμήμα 3x3 της εικόνας που θέλουμε να κάνουμε συνέλιξη αυτήν θα κάνει τους πολλαπλασιασμούς μεταξύ των στοιχείων των δυο πινάκων και θα αθροίσει το αποτέλεσμα επιστρέφοντας έναν αριθμό για το συγκεκριμένο βήμα. Θα καλείτε μέσα στην Conv3D για ταυτόχρονο υπολογισμό της συνέλιξης σε 3 διαδοχικά frame .

Ο κώδικας φαίνεται στον παρακάτω πίνακα :

```
def conv_single_step(slice, filt):
    s = np.multiply(slice,filt)
    Z = np.sum(s)
    return Z
```

- **myConv3D(A, B ,param) :**

Η συνάρτηση αυτήν είναι η σημαντικότερη καθώς εφαρμόζει την 3D συνέλιξη στο βίντεο. Δέχεται σαν είσοδο 150 gray scale frames , το φίλτρο και την παράμετρο param. Αυτό που κάνει η συνάρτηση σε ένα round είναι να σκαναρει κατά τους χωρικούς άξονες X και Y σε 3 frame ταυτόχρονα κάθε φορά . Χρησιμοποιούμε 3 for , η πρώτη για να περνούμε τα frame και οι άλλες 2 για να σκαναρουμε κατά τους άξονες X και Y. Η έξω for θα τρέξει 148 φορές όσα και τα frame εξόδου , η Conv3D καλεί την conv_single_step 3 φορές για κάθε ένα 1 από τα 3 frame που χρησιμοποιούμε σε ένα round και αθροίζει το άθροισμα τους κάθε φορά καλωντας την conv_single_step . Η συνάρτηση πρέπει να να επιστρέφει πίνακα διάστασης ίσης με τον πίνακα A αυτό συμβαίνει όταν έχουμε «Same Conv» δηλαδή έχουμε pad≠0 και συγκριμένα το $\text{pad} = \frac{f-1}{2} = 1$ για 3x3 kernel, $\text{pad} = \frac{f-1}{2} = 2$ για 5x5 κλπ. , στην «Valid Conv» έχουμε pad=0. Η συνάρτηση pad_image φροντίζει για τον υπολογισμό του σωστού pad σύμφωνα με τον παραπάνω τύπο , Επομένως ελέγχω το pad στην συνάρτηση και αν είναι "same" καλώ την pad_image γιατί στην 3D συνέλιξη σε βίντεο δεν θέλω να έχω "Valid Conv" κάτι που είναι αρκετά χρήσιμο στα CNN .

Ο κώδικας φαίνεται στον παρακάτω πίνακα :

```
def myConv3D(A, B ,param):
    checkparam = param["same"]
    nH=A.shape[1]
    nW=A.shape[2]
    f=B.shape[0]
    m=A.shape[0]
    pad = 1
    stride = 1
    nH_new = int((nH-f+2*pad)/stride)+1
    nW_new = int((nW-f+2*pad)/stride)+1
    conv_output_3D = np.zeros((m-2, nH_new, nW_new))
    if checkparam==1 :
        A_pad_image = pad_image(A, f)
    else:
        print("be careful after padding we dont have same dimensions with matrix A !")
    for i in range(m-2):
        for h in range(nH_new):
            vertical_start = stride*h
            vertical_end = stride*h+f
            for w in range(nW_new):
                horizontal_start = stride*w
                horizontal_end = stride*w+f
                a_slice_1 = A_pad_image[i, vertical_start:vertical_end, horizontal_start:horizontal_end]
                a_slice_2 = A_pad_image[i+1, vertical_start:vertical_end, horizontal_start:horizontal_end]
                a_slice_3 = A_pad_image[i+2, vertical_start:vertical_end, horizontal_start:horizontal_end]
                conv_output_3D[i,h,w] =conv_single_step(a_slice_1,B[0])
                                       +conv_single_step(a_slice_2,B[1])
                                       +conv_single_step(a_slice_3,B[2])
    return conv_output_3D
```

- **main() :**

Η main ελέγχει όλες τις συναρτήσεις αρχικά δημιουργεί το φίλτρο , στην συνέχεια μετατρέπει το βίντεο σε ασπρόμαυρο . Δημιουργεί έναν πυρήνα διαστάσεων 3x3x3 κάνοντας χρήση της αντίστοιχης συνάρτησης. Εισάγει zero padding στο βίντεο. Εφαρμόζει συνέλιξη του βίντεο με τον πυρήνα . Αποθηκεύει το αποτέλεσμα της συνέλιξης με την μορφή 'mp4' .

Ο κώδικας φαίνεται στον παρακάτω πίνακα :

```
def main():
    start_time = time.time()
    kernel=create_smooth_kernel(3)
    print("kernel.shape =\n",kernel.shape)
    param = {"same" : 1, "valid": 0}
    # Extract frames from Video & grayscale transform
    cap = cv2.VideoCapture('video.mp4')
    frame_Count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_Width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_Height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    buf = np.empty((frame_Count, frame_Height, frame_Width), np.dtype('uint8'))
    fc = 0
    while (fc < frame_Count):
       ret, img = cap.read()
       buf[fc]  = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
       fc += 1
    cap.release()
    cv2.destroyAllWindows()
    # Calculate 3D convolution
    conv_output_3D = myConv3D(buf, kernel,param)
    # Writing video after 3D Convolution
    conv_output_3D = conv_output_3D.astype(np.uint8)
    skvideo.io.vwrite("videofinal.mp4", conv_output_3D)
    print("Total running time : %s seconds " % (time.time() - start_time))
```

Παραδοτέα :

1. 3D_convolution_George_Kalitsios.py
2. Videofinal.mp4
3. Report_Advanced_CV_Programming_Exercise_2020_Kalitsios_Georgios.pdf