extraction/traitement_3D.py

```
##3D traitement

#importations
import os
import numpy as np
import pickle

from vtk import vtkDecimatePro, vtkPolyDataWriter, vtkPolyDataReader,
vtkPolyData


def decimate(vtk_file, ratio):
    reader = vtkPolyDataReader()
    reader.SetFileName(vtk_file)
    reader.Update()
    input_poly = reader.GetOutput()
    decim_poly = vtkPolyData()

    decimator = vtkDecimatePro()
    decimator.SetInputData(input_poly)
    decimator.SetTargetReduction(ratio)
    decimator.Update()

    decim_poly.ShallowCopy(decimator.GetOutput())

    writer = vtkPolyDataWriter()
    writer.SetInputData(decim_poly)
    #writer.SetFileName("_decimated.".join(vtk_file.split(".")))
    writer.SetFileName(vtk_file)
    #writer.Update()
    writer.Write()



def genimage(image_name, colour, image_modele):
    command("genimage "+image_modele+" "+str(colour)+" "+image_name)
    return image_name

def pgm2ppm(red, green, blue, image_out):
    command("pgm2ppm "+red+" "+green+" "+blue+" "+image_out)
    return image_out

def into_ppm(image_pgm, image_out):
    return pgm2ppm(image_pgm, image_pgm, image_pgm, image_out)

def add(image_a, image_b, image_out):
    command("add "+image_a+" "+image_b+" "+image_out)
    return image_out

def sub(image_a, image_b, image_out):
    command("sub "+image_a+" "+image_b+" "+image_out)
    return image_out

def extractplane(image_3D, plane, couche, image_out):
    command("extractplane "+image_3D+" "+str(couche)+" "+plane+"
"+image_out)
    return image_out
```

```
def drawball(image_in, radius, x, y, z, image_out):
    command("drawball "+image_in+" "+str(rayon)+" "+str(x)+" "+str(y)+"
"+str(z)+" "+image_out)
    return image_out

def extrait_coupes(image_3D, plane, debut, fin, prefixe):
    n = int(np.log(fin)) -1
    for i in range(debut, fin):
        extractplane(image_3D, plane, i, prefixe+numerote(i, n))


def inverse(image_in, image_out):
    command("inverse "+image_in+" "+image_out)
    return image_out

def colorie(image_pgm, color, image_out):
    tmp_name = "TMP_colorie_image_mienglknzarmkazemkfanzzaf"
    aplat_rouge = genimage(tmp_name+"aR", 255-color[0], image_pgm)
    aplat_vert  = genimage(tmp_name+"aG", 255-color[1], image_pgm)
    aplat_bleu  = genimage(tmp_name+"aB", 255-color[2], image_pgm)
    rouge = sub(image_pgm, aplat_rouge, aplat_rouge+ "s")
    vert  = sub(image_pgm, aplat_vert , aplat_vert + "s")
    bleu  = sub(image_pgm, aplat_bleu , aplat_bleu + "s")
    rgb = pgm2ppm(rouge, vert, bleu, image_out)
    os.system("rm "+aplat_rouge)
    os.system("rm "+aplat_vert)
    os.system("rm "+aplat_bleu)
    os.system("rm "+rouge)
    os.system("rm "+vert)
    os.system("rm "+bleu)
    return rgb

def frame(image_modele, image_out):
    command("frame "+image_modele+" "+image_out)
    return image_out

def geodilat(image_in, mask, connexity, n_iterations, image_out):
    command("geodilat "+image_in+" "+mask+" "+str(connexity)+"
"+str(n_iterations)+" "+image_out)
    return image_out

## l'image doit etre binaire
def vire_bord(image_in, image_out):
    fr = frame(image_in, "TMP_frame_azljebagambmqojbbbljzab")
    dil = geodilat(fr, image_in, 26, -1, "TMP_frame_ajnmvknapzrnoabnp")
    os.system("rm "+fr)
    return sub(image_in, dil, image_out)


#donnees
n_tempo = 16 # nb de temps
n_coupes_xy = 250 # dimensions des colones en nombres de coupes
n_coupes_xz = 80
n_coupes_yz = 80
seuil = 170 #pour les images en noir et blanc

#appel de fc de pink
def command(cmd):
    os.system("../pink/linux/bin/"+cmd)
```

```
#fonction pour numeroter les fichiers
def numerote(n, l):
   s = str(n)
   while len(s) < l:
       s = '0' + s
   return s

#ajoute le volume a la lise des barycentres
def add_volume(file_name, image_name):
    f = open(file_name, "r")
    lines = f.readlines();
    f.close()
    f = open(file_name, "w")
    f.write(lines[0])
    f.close()
    for i in range(1, len(lines)):
        command("showpoint "+image_name+" "+lines[i][:-1]+" |
trucs_en_c/filtre_showpoint_3D >> " + file_name)


#recupere les elements d un .list
def parse_list(file_name):
    f = open(file_name, "r")
    line = f.readline()
    mode = line[0]
    n_els = int(line[2:][:-1])
    liste = []
    for i in range(n_els):
        liste.append([])
        line = f.readline()[:-1];
        for s in line.split(" "):
            liste[i].append(int(s))
    f.close()
    return liste


def export_list(liste, file_name):
    os.system("rm "+file_name)
    f = open(file_name, "w")
    f.write("_ "+str(len(liste))+"\n")
    for elements in liste:
        f.write(" ".join(str(el) for el in elements)+"\n")
    f.close()

def traitement_3D_main( fichierDemandeParUtilisateur = "gros_sable.tif",
supprimerBords=False ):

    #palette de couleurs (pour generer la carte des distances coloree)
    lut=" lut "
    lut_bord = " lut_bord "
    command("genlut 1000 0 1 0 200 200 200"+lut)
    command("genlut 256 0 0 0 255 255 255"+lut_bord)


    #gestion des repertoires

    os.system("rm -R tmp")
    os.system("mkdir tmp")

    os.system("rm -R images_3D")
    os.system("mkdir images_3D")
```

```
    os.system("rm -R vtk_3D")
    os.system("mkdir vtk_3D")

    os.system("rm -R images_2D_for_3D")
    os.system("mkdir images_2D_for_3D")

    os.system("rm -R coupes_3D")
    os.system("mkdir coupes_3D")

    os.system("mkdir coupes_3D/originales")
    os.system("mkdir coupes_3D/originales/x_y")
    os.system("mkdir coupes_3D/originales/x_z")
    os.system("mkdir coupes_3D/originales/y_z")

    os.system("mkdir coupes_3D/carte_dist")
    os.system("mkdir coupes_3D/carte_dist/x_y")
    os.system("mkdir coupes_3D/carte_dist/x_z")
    os.system("mkdir coupes_3D/carte_dist/y_z")

    os.system("mkdir coupes_3D/contours_blancs")
    os.system("mkdir coupes_3D/contours_blancs/x_y")
    os.system("mkdir coupes_3D/contours_blancs/x_z")
    os.system("mkdir coupes_3D/contours_blancs/y_z")

    os.system("mkdir coupes_3D/contours_rouges")
    os.system("mkdir coupes_3D/contours_rouges/x_y")
    os.system("mkdir coupes_3D/contours_rouges/x_z")
    os.system("mkdir coupes_3D/contours_rouges/y_z")

    os.system("mkdir coupes_3D/borders")
    os.system("mkdir coupes_3D/borders/x_y")
    os.system("mkdir coupes_3D/borders/x_z")
    os.system("mkdir coupes_3D/borders/y_z")

    os.system("mkdir coupes_3D/water")
    os.system("mkdir coupes_3D/water/x_y")
    os.system("mkdir coupes_3D/water/x_z")
    os.system("mkdir coupes_3D/water/y_z")

    os.system("rm -R border_3D")
    os.system("mkdir border_3D")


    os.system("rm -R labels_3D")
    os.system("mkdir labels_3D")

    os.system("rm -R bary_3D")
    os.system("mkdir bary_3D")
    os.system("mkdir bary_3D/liste")

    os.system("rm -R volumes_3D")
    os.system("mkdir volumes_3D")

    #extraction des images du tif en bmp
    os.system("convert " + fichierDemandeParUtilisateur + " tmp/test.bmp")

    #conversion en pgm et renommage aux standard de la fc catpgm
    for i in range(0, n_tempo*n_coupes_xy):
        os.system("convert tmp/test-"+str(i)+".bmp
images_2D_for_3D/image_2D_for_3D_"+numerote(i,4)+".pgm")
```

```
    #traitement pour chaque temps

    for t in range(0, n_tempo):

        print("image 3D numero "+str(t)+" en traitement")

        debut_colone = t * n_coupes_xy
        fin_colone   = debut_colone + n_coupes_xy - 1

        padding_temporel = numerote(t, 2)


        #generation de l image 3D a partir des 250 coupes
        command("catpgm images_2D_for_3D/image_2D_for_3D_ "+
                str(debut_colone)+" "+str(fin_colone)+
                " [80 80 250]
images_3D/image_3D_t"+padding_temporel+".pgm")


        #seuillage
        command("seuil images_3D/image_3D_t"+padding_temporel+".pgm "+
                str(seuil)+" "+
                "images_3D/image_3D_s_t"+padding_temporel+".pgm")


        #nettoyage des impurtes dans les grains
        file_to_clean = " images_3D/image_3D_s_t"+padding_temporel+".pgm "

        minimum_noir = "300"
        minimum_blanc = "30"

        command("attribute"+file_to_clean+"6 "+minimum_noir+" 1 0
attributes")
        command("long2byte attributes attributes")
        command("attribute attributes 6 "+minimum_blanc+" 1 0 attributes")
        command("long2byte attributes attributes")
        command("seuil attributes 1 "+file_to_clean)


        command("inverse images_3D/image_3D_s_t"+padding_temporel+".pgm "+
                "images_3D/image_3D_s_inv_t"+padding_temporel+".pgm")


         # carte des distances, avec distance euclidienne
        command("dist images_3D/image_3D_s_inv_t"+padding_temporel+".pgm "+
                "0 images_3D/image_3D_dist_proc_t"+padding_temporel+".pgm")

        command("long2byte
images_3D/image_3D_dist_proc_t"+padding_temporel+".pgm "+
                "0 images_3D/image_3D_dist_t"+padding_temporel+".pgm")

        command("inverse images_3D/image_3D_dist_t"+padding_temporel+".pgm
"+
                "images_3D/image_3D_dist_inv_t"+padding_temporel+".pgm")


        # minimas
        command("minima
images_3D/image_3D_dist_inv_t"+padding_temporel+".pgm "+
```

```
                "6 images_3D/image_3D_min_t"+padding_temporel+".pgm")


        #ligne de separation des eaux
        command("watershed
images_3D/image_3D_dist_inv_t"+padding_temporel+".pgm "+
                "images_3D/image_3D_min_t"+padding_temporel+".pgm "+
                "26 images_3D/image_3D_wat_t"+padding_temporel+".pgm")

        command("add images_3D/image_3D_s_inv_t"+padding_temporel+".pgm "+
                "images_3D/image_3D_wat_t"+padding_temporel+".pgm "+
                "images_3D/image_3D_superpose_t"+padding_temporel+".pgm")


        image_supinv =  "
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm "

        command("inverse
images_3D/image_3D_superpose_t"+padding_temporel+".pgm " + image_supinv)

        if supprimerBords:
            vire_bord(image_supinv, image_supinv)


        #creation du vtk
        lissage = 5
        nom_vtk = "vtk_3D/vtk_t_"+padding_temporel+".vtk"
        command("mcube
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm 0
"+str(lissage)+" 0 VTK "+nom_vtk)
        print("decimating", nom_vtk)
        decimate(nom_vtk, .9)
        print("done")
        #border
        command("border
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm 26
border_3D/border_3D_t_"+padding_temporel+".pgm")
        command("add images_3D/image_3D_t"+padding_temporel+".pgm
border_3D/border_3D_t_"+padding_temporel+".pgm
border_3D/border_3D_t_"+padding_temporel+"_add_blanc.pgm")


        #barycentres

        command("3dlabel
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm
labels_3D/label_t_"+padding_temporel+".pgm")
        command("barycentrelab labels_3D/label_t_"+padding_temporel+".pgm
bary_3D/bary_3D_t"+padding_temporel+".pgm")
        command("pgm2list bary_3D/bary_3D_t"+padding_temporel+".pgm B
bary_3D/liste/bary_list_t"+padding_temporel+".list")


        #volumes
        command("attribute
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm 26 0 0 0
volumes_3D/volume_3D_t_"+padding_temporel+".pgm")

add_volume("bary_3D/liste/bary_list_t"+padding_temporel+".list","volumes_3D
/volume_3D_t_"+padding_temporel+".pgm")
```

```
        #extraction des coupes 2D sur les 3 plans

        # pour moins de repetitions
        debut_commande= "extractplane
border_3D/border_3D_t_"+padding_temporel+"_add_blanc.pgm "



        #extraction des coupes sur (x,y)

        os.system("mkdir coupes_3D/originales/x_y/"+padding_temporel)
        os.system("mkdir coupes_3D/borders/x_y/"+padding_temporel)
        os.system("mkdir coupes_3D/water/x_y/"+padding_temporel)
        os.system("mkdir coupes_3D/carte_dist/x_y/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_rouges/x_y/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_blancs/x_y/"+padding_temporel)



        for i_coupe in range(0, n_coupes_xy):

            #image originale
            command("extractplane
images_3D/image_3D_t"+padding_temporel+".pgm "+str(i_coupe)+" xy "+

"coupes_3D/originales/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xy_"+numerote(i_coupe,4)+"_org.pgm")


            #carte des distances coloree
            command("extractplane
images_3D/image_3D_dist_proc_t"+padding_temporel+".pgm "+str(i_coupe)+" xy
"+

"coupes_3D/carte_dist/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xy_"+numerote(i_coupe,4)+"_dist.pgm")
            command("colorize
coupes_3D/carte_dist/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_x
y_"+numerote(i_coupe,4)+"_dist.pgm"

+lut+"coupes_3D/carte_dist/x_y/"+padding_temporel+"/t_"+padding_temporel+"c
oupe_xy_"+numerote(i_coupe,4)+"_distcolor.ppm")


            #watershed fini
            command("extractplane
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm "+str(i_coupe)+"
xy "+

"coupes_3D/water/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"+
numerote(i_coupe,4)+"_wa.pgm")

            #borders
            command("border
coupes_3D/water/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"+n
umerote(i_coupe,4)+
                    "_wa.pgm 8
coupes_3D/borders/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"
+numerote(i_coupe,4)+"_border.pgm")
```

```
            #image originale contours rouges
            command("colorize
coupes_3D/borders/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"
+numerote(i_coupe,4)+"_border.pgm"+lut_bord+

"coupes_3D/borders/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_
"+numerote(i_coupe,4)+"_border_rg.ppm")

            image_a_ppm =
"coupes_3D/originales/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xy_"+numerote(i_coupe,4)+"_org.pgm"

            command("pgm2ppm "+image_a_ppm+" "+image_a_ppm+" "+image_a_ppm+
                    "
coupes_3D/contours_rouges/x_y/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xy_"+numerote(i_coupe,4)+"_orgppm.ppm")

            command("sub
coupes_3D/contours_rouges/x_y/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xy_"+numerote(i_coupe,4)+
                    "_orgppm.ppm
coupes_3D/borders/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"
+numerote(i_coupe,4)+
                    "_border_rg.ppm
coupes_3D/contours_rouges/x_y/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xy_"+numerote(i_coupe,4)+"_controuge.ppm")



            #image originale contours blancs
            command("add
coupes_3D/originales/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_x
y_"+numerote(i_coupe,4)+
                    "_org.pgm
coupes_3D/borders/x_y/"+padding_temporel+"/t_"+padding_temporel+"coupe_xy_"
+numerote(i_coupe,4)+
                    "_border.pgm
coupes_3D/contours_blancs/x_y/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xy_"+numerote(i_coupe,4)+"_contblanc.pgm")




        #extraction des coupes sur (x,z)

        os.system("mkdir coupes_3D/originales/x_z/"+padding_temporel)
        os.system("mkdir coupes_3D/borders/x_z/"+padding_temporel)
        os.system("mkdir coupes_3D/water/x_z/"+padding_temporel)
        os.system("mkdir coupes_3D/carte_dist/x_z/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_rouges/x_z/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_blancs/x_z/"+padding_temporel)

        for i_coupe in range(0, n_coupes_xz):
            #image originale
            command("extractplane
images_3D/image_3D_t"+padding_temporel+".pgm "+str(i_coupe)+" xz "+
```

```
"coupes_3D/originales/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xz_"+numerote(i_coupe,4)+"_org.pgm")


            #carte des distances coloree
            command("extractplane
images_3D/image_3D_dist_proc_t"+padding_temporel+".pgm "+str(i_coupe)+" xz
"+

"coupes_3D/carte_dist/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xz_"+numerote(i_coupe,4)+"_dist.pgm")
            command("colorize
coupes_3D/carte_dist/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_x
z_"+numerote(i_coupe,4)+"_dist.pgm"

+lut+"coupes_3D/carte_dist/x_z/"+padding_temporel+"/t_"+padding_temporel+"c
oupe_xz_"+numerote(i_coupe,4)+"_distcolor.ppm")


            #watershed fini
            command("extractplane
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm "+str(i_coupe)+"
xz "+

"coupes_3D/water/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"+
numerote(i_coupe,4)+"_wa.pgm")

            #borders
            command("border
coupes_3D/water/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"+n
umerote(i_coupe,4)+
                    "_wa.pgm 8
coupes_3D/borders/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"
+numerote(i_coupe,4)+"_border.pgm")



            #image originale contours rouges
            command("colorize
coupes_3D/borders/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"
+numerote(i_coupe,4)+"_border.pgm"+lut_bord+

"coupes_3D/borders/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_
"+numerote(i_coupe,4)+"_border_rg.ppm")

            image_a_ppm =
"coupes_3D/originales/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
xz_"+numerote(i_coupe,4)+"_org.pgm"

            command("pgm2ppm "+image_a_ppm+" "+image_a_ppm+" "+image_a_ppm+
                    "
coupes_3D/contours_rouges/x_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xz_"+numerote(i_coupe,4)+"_orgppm.ppm")

            command(
                "sub
coupes_3D/contours_rouges/x_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xz_"+numerote(i_coupe,4)+"_orgppm.ppm "+
```

```
                "
coupes_3D/borders/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"
+numerote(i_coupe,4) + "_border_rg.ppm "+
                "
coupes_3D/contours_rouges/x_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xz_"+numerote(i_coupe,4)+"_controuge.ppm"
            )


            #image originale contours blancs
            command("add
coupes_3D/originales/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_x
z_"+numerote(i_coupe,4)+
                    "_org.pgm
coupes_3D/borders/x_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_xz_"
+numerote(i_coupe,4)+
                    "_border.pgm
coupes_3D/contours_blancs/x_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_xz_"+numerote(i_coupe,4)+"_contblanc.pgm")



        #extraction des coupes sur (y,z)

        os.system("mkdir coupes_3D/originales/y_z/"+padding_temporel)
        os.system("mkdir coupes_3D/borders/y_z/"+padding_temporel)
        os.system("mkdir coupes_3D/water/y_z/"+padding_temporel)
        os.system("mkdir coupes_3D/carte_dist/y_z/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_rouges/y_z/"+padding_temporel)
        os.system("mkdir coupes_3D/contours_blancs/y_z/"+padding_temporel)

        for i_coupe in range(0, n_coupes_yz):

            #image originale
            command("extractplane
images_3D/image_3D_t"+padding_temporel+".pgm "+str(i_coupe)+" yz "+

"coupes_3D/originales/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
yz_"+numerote(i_coupe,4)+"_org.pgm")


            #carte des distances coloree
            command("extractplane
images_3D/image_3D_dist_proc_t"+padding_temporel+".pgm "+str(i_coupe)+" yz
"+

"coupes_3D/carte_dist/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
yz_"+numerote(i_coupe,4)+"_dist.pgm")
            command("colorize
coupes_3D/carte_dist/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_y
z_"+numerote(i_coupe,4)+"_dist.pgm"

+lut+"coupes_3D/carte_dist/y_z/"+padding_temporel+"/t_"+padding_temporel+"c
oupe_yz_"+numerote(i_coupe,4)+"_distcolor.ppm")


            #watershed fini
            command("extractplane
images_3D/image_3D_superpose_inv_t"+padding_temporel+".pgm "+str(i_coupe)+"
yz "+
```

```
"coupes_3D/water/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"+
numerote(i_coupe,4)+"_wa.pgm")

            #borders
            command("border
coupes_3D/water/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"+n
umerote(i_coupe,4)+
                    "_wa.pgm 8
coupes_3D/borders/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"
+numerote(i_coupe,4)+"_border.pgm")



            #image originale contours rouges
            command("colorize
coupes_3D/borders/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"
+numerote(i_coupe,4)+"_border.pgm"+lut_bord+

"coupes_3D/borders/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_
"+numerote(i_coupe,4)+"_border_rg.ppm")

            image_a_ppm =
"coupes_3D/originales/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_
yz_"+numerote(i_coupe,4)+"_org.pgm"

            command("pgm2ppm "+image_a_ppm+" "+image_a_ppm+" "+image_a_ppm+
                    "
coupes_3D/contours_rouges/y_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_yz_"+numerote(i_coupe,4)+"_orgppm.ppm")

            command("sub
coupes_3D/contours_rouges/y_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_yz_"+numerote(i_coupe, 4)+
                    "_orgppm.ppm
coupes_3D/borders/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"
+numerote(i_coupe, 4)+
                    "_border_rg.ppm
coupes_3D/contours_rouges/y_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_yz_"+numerote(i_coupe, 4)+"_controuge.ppm")



            #image originale contours blancs
            command("add
coupes_3D/originales/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_y
z_"+numerote(i_coupe, 4)+
                    "_org.pgm
coupes_3D/borders/y_z/"+padding_temporel+"/t_"+padding_temporel+"coupe_yz_"
+numerote(i_coupe, 4)+
                    "_border.pgm
coupes_3D/contours_blancs/y_z/"+padding_temporel+"/t_"+padding_temporel+"co
upe_yz_"+numerote(i_coupe, 4)+"_contblanc.pgm")




    #listage des grains

    #enleve les formes qui ne sont pas des grains de la liste des barry
```

```
    formes=[]

    for t in range(0, n_tempo):

        padding_temporel = numerote(t, 2) # pour pas trop recalculer

        #enleve de la liste les grains de vol<100
        liste=
parse_list("../extraction/bary_3D/liste/bary_list_t"+padding_temporel+".lis
t")
        volume_min=100
        i_grain=0
        while i_grain< len(liste):
            if (liste[i_grain][3]<volume_min):
                del(liste[i_grain])
            else:
                i_grain += 1

        formes.append(liste)


    grains = []

    # distance entre 2 grains
    def dist2(g1, g2):
        return pow(g1[0]-g2[0], 2) + pow(g1[1]-g2[1], 2) + pow(g1[2]-g2[2],
2)


    vol_moy=0
    n_grains=0
    for  time in formes:
        for grain in time:
            vol=grain[3]
            if (vol>7000):
                vol_moy=vol_moy+grain[3]
                n_grains=n_grains+1

    vol_moy=vol_moy/n_grains

    seuil_volume = 100

    #resultats = []
    #grains = []

    vitesses=[]


    num_grain=0
    vitesse_moy_grains=0
    acc_moy_grains=0


    for grain in formes[0]:
        vitesse=0
        acc=0
        grains.append(
            [grain[3],
             [grain[:3]]
            ])
```

```
        for t in range(1, n_tempo):
            gr_prec = grains[len(grains)-1][1][t-1]
            dist_min = dist2(gr_prec, formes[t][0])
            plus_proche = formes[t][0]

            for suivant in formes[t]:
                d2 = dist2(gr_prec, suivant)
                if (d2 < dist_min and abs(gr_prec[0]-suivant[0]) <
seuil_volume):
                    dist_min = d2
                    plus_proche = suivant
            grains[len(grains)-1][1].append(plus_proche[:3])


            #vitesse
            vitesse=0
            vx = abs(grains[num_grain][1][t][0]-grains[num_grain][1][t-
1][0])
            vy = abs(grains[num_grain][1][t][1]-grains[num_grain][1][t-
1][1])
            vz = abs(grains[num_grain][1][t][2]-grains[num_grain][1][t-
1][2])
            v= (vx**2+vy**2+vz**2)**(0.5)
            vitesse=vitesse+v

            if (t>1):
                 vx_1 = abs(grains[num_grain][1][t-1][0]-
grains[num_grain][1][t-2][0])
                 vy_1 = abs(grains[num_grain][1][t-1][1]-
grains[num_grain][1][t-2][1])
                 vz_1 = abs(grains[num_grain][1][t-1][2]-
grains[num_grain][1][t-2][2])
                 v_1= (vx_1**2+vy_1**2+vz_1**2)**(0.5)
                 a=abs(v-v_1)
                 acc=acc+a

        acc=acc/13
        vitesse = vitesse/14
        vitesses.append([grains[num_grain][1][0],vitesse,acc])
        vitesse_moy_grains = vitesse_moy_grains + vitesse
        acc_moy_grains = acc_moy_grains+ acc


        num_grain=num_grain+1

    nb_grains=len(grains)

    #vitesse moyenne des grains
    vitesse_moy_grains=vitesse_moy_grains/nb_grains

    #acceleration moyenne des grains
    acc_moy_grains=acc_moy_grains/nb_grains

    np.save("../extraction/tracking_3D/vitesse_moy_grains.npy",
[vitesse_moy_grains,acc_moy_grains,nb_grains,vol_moy])
    np.save("../extraction/tracking_3D/grains.npy", grains)
    np.save("../extraction/tracking_3D/vitesses.npy",vitesses)


    resultats = []
    for grain in grains:
```

```
            x = []
            y = []
            z = []
            for coords in grain[1]:
                x.append(coords[0])
                y.append(coords[1])
                z.append(coords[2])
            resultats.append([x, y, z])

        np.save("../extraction/tracking_3D/resultats.npy", resultats)



        NB_IMGS = n_tempo * n_coupes_xy
        INTERVALLE_XY = n_coupes_xy
        INTERVALLE_XZ = n_coupes_xz
        INTERVALLE_YZ = n_coupes_yz
        URL_PGM = "../extraction/coupes_3D/"
        URL_VTK = "../extraction/vtk_3D/"
        URL_GRAPHIQUE_3D = "../extraction/tracking_3D/resultats.npy"
        URL_VITESSE_MOY_GRAINS =
"../extraction/tracking_3D/vitesse_moy_grains.npy"
        """
        EXPORTATION
        """
        #base de donnees
        bdd = {}
        bdd["NB_IMGS"] = NB_IMGS
        bdd["INTERVALLE_XY"] = INTERVALLE_XY
        bdd["INTERVALLE_XZ"] = INTERVALLE_XZ
        bdd["INTERVALLE_YZ"] = INTERVALLE_YZ
        bdd["URL_PGM"] = URL_PGM
        bdd["URL_VTK"] = URL_VTK
        bdd["URL_GRAPHIQUE_3D" ] = URL_GRAPHIQUE_3D
        bdd["URL_VITESSE_MOY_GRAINS" ] = URL_VITESSE_MOY_GRAINS

        # Its important to use binary mode
        fichierBdd = open('woopwoop', 'wb')

        # source, destination
        pickle.dump(bdd, fichierBdd)
        fichierBdd.close()


        return os.path.abspath('woopwoop')
```