REPRESENTATION DE L'INFORMATION
Partie 1 : Représentation des images

Document enseignant

1 Présentation :

Le contenu de ce chapitre a été fait selon le paragraphe §4.1, la représentation de l'information, du programme officiel de la spécialité ISN. Dans ce papier, dans un premier temps on étudie le cas des images en se limitant dans le cas simple d'image non compressé. Puis, en application, un traitement de conversion d'image 3D au format SBS en anaglyphe.

- Représentation de l'information : image
- Algorithmique : réflexion sur le traitement
- Langage : python (programme) (amélioration avec la commande shell bash)
- Architecture matériel : Intel 486 – OS : Linux (exemple Clef[i] USB ISN)

2 Représentation de l'image[ii] :

Une méthode pour numériser une image est de la quadriller. Chacune des cases du quadrillage est un élément de base de l'image appelé pixel. Cet élément prend la teinte de l'image en ce point.
Si l'image est en noir et blanc, il ne prend que deux valeurs, noir ou blanc (ou du binaire en numérique, soit 1 ou 0). En lisant ces pixels ligne par ligne, cela revient à les placer sous forme d'une suite de valeurs, cette méthode est le bitmap.

3 Image au format bitmap :

Pour stocker ces informations, on peut les placer dans un fichier de données. Ici on parle de format pour indiquer le rangement de ces informations dans le fichier.
Un des formats le plus simple est le PBM (Portable BitMap). Un fichier texte en ASCII contenant les en-têtes (informations sur l'image) et la suite de pixels.
Exemple d'un contenu de fichier PBM en noir et blanc :

```
P1
# mon image en fichier PBM : image001 en NB
24 12
000000000000000000000000
000000000000000000000010
001100001100000000000000
000010011000000000000000
000000000000000000000000
000001000000000000000000
000001100000000000000000
001000000010000000000000
000110001100000000000000
000001110000000000000000
000000000000000000000000
000000000000000000000000
```

donne 

avec :

- Le caractère P1, suivi d'un espace ou retour chariot (indiquant le type d'image PBM en NB) ;
- La largeur de l'image en base 10, suivi d'espace ou de retour chariot ;
- La hauteur de l'image en base 10, suivi d'espace ou de retour chariot ;
- La suite de pixels ligne par ligne ; de gauche à droite, de haut en bas.

4 Exercice :

Analyses documentaires et recherche personnelles de l'élève : à partir des informations de votre livre, créer un fichier pbm représentant le drapeau tricolore.

Partie 2 : Application en traitement d'image 3D
(SBS en anaglyphe)

Document enseignant

1 Pré-requis :

La partie 1 ci-dessus.

2 Principe de la projection d'image en 3D

Lorsqu'on regarde un objet l'œil gauche ne voit pas la même chose que l'œil droit. Ces deux informations sont traitées par le cerveau qui nous donnent la notion d'espace de vision. La projection d'image 3D sur un écran 2D consiste à reconstituer ce principe.
Les sources de données (ici, les images) sont une paire d'images, gauche et droite. Elles sont souvent stockées ensemble. Les méthodes d'assemblage de ses deux images peuvent être soit dessus-dessous (dite Up-Down ou UD) soit face-à-face (dite Side by Side ou SBS). Les formats de stockage peuvent se faire de la même manière qu'en bitmap.
Les images de sortie :
cas 1 : La sortie est une image double en SBS si l'observateur possède une paire de lunettes du type parallèle ou SBS inversée croisée du type cross-over.
cas 2 : La sortie est une image multiplexée dite anaglyphe. Ses composantes en couleur soit RougeDroite, VertGauche et BleueGauche soit RougeGauche, VertDroite et BleueDroite. Le choix dépend de la paire de lunettes de visualisation ; une RougeBleue ou BleueRouge.

3 Cahier de charge :

- Faire l'algorithme et écrire un programme de traitement d'image pour convertir **une** photo SBS en **une** photo anaglyphe (rouge-cyan).
- Langage de programmation : python – (on peut créer un lanceur externe par le shell en bash pour automatiser le traitement de plusieurs photos.)

à faire :

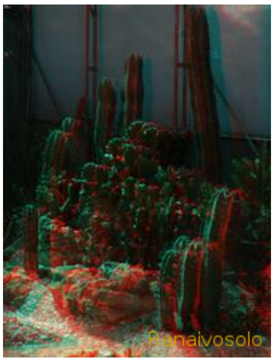
- on importe les librairies utiles au traitement
- charger l'image source laSource
- récupérer sa taille
- créer deux images tampons pourLaGauche et pourLaDroite
- extraire ces images pourLaGauche et pourLaDroite depuis laSource
- créer deux images de sorties RBsortie et Brsortie (pour les cas 2 cité en haut)
- sauvegarder les (éventuellement les visualiser).

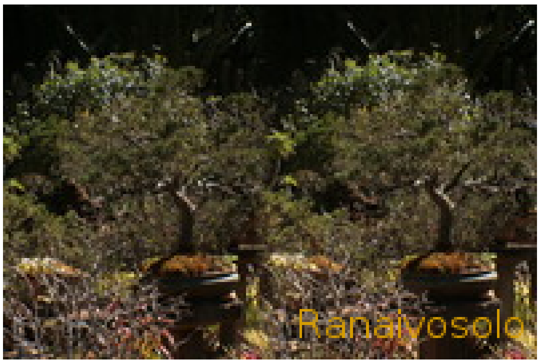
4 Les lunettes 3D non polarisé : Pour visionner les images il vous faut ceux-ci

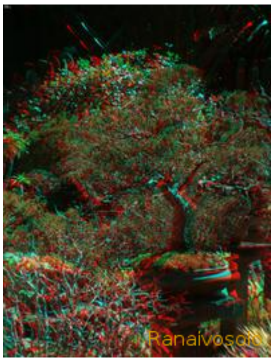
pour photo en SBS

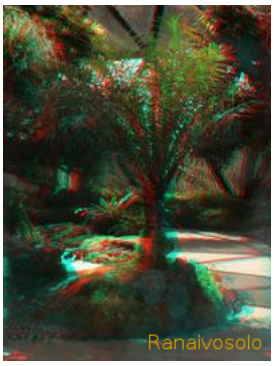
pour photo en anaglyphe

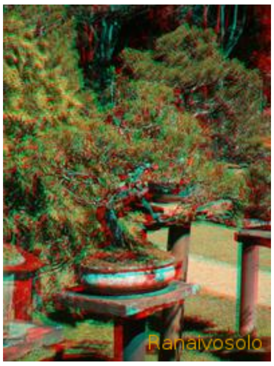
5 Quelques photos traitées

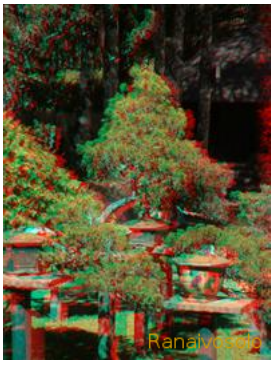


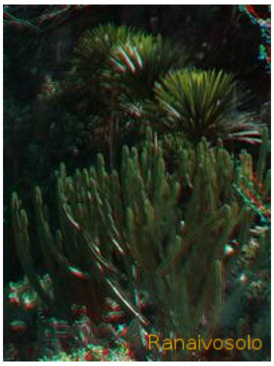







Ranaivosolo

Ranaivosolo

Ranaivosolo

Ranaivosolo

Ranaivosolo

Ranaivosolo

Ranaivosolo

Ranaivosolo

6 Code de programme :

Ce code a été testé sous slackware 14.0.

```
#!/usr/bin/python
# pour python 2.x
# Conversion d'une imabe Side by Side en Anaglyphe Rouge-Cyan
import os, sys;
import Image, ImageDraw ;
# initialisation
bon = True;
nomFichier="img_2937.jpg"
ptG=[];
ptD=[];

# un point commun
tailleSortie=[];
tailleSortie.append(0);
tailleSortie.append(0);
print;
# lecture du fichier image
try:
   im = Image.open(nomFichier);
except IOError:
   print "Fichier illisible";
   print "Changer le nom de fichier et reessayer ;-) ";
   print
   bon = False ;
if bon :
   print "Bonjour !"
   print "Fichier lu : ",nomFichier
   print
   myTuple=im.size;
   L=im.size[0];
   H=im.size[1];
   print "format : ",im.format, im.size, im.mode;
   draw = ImageDraw.Draw(im)
   draw.text((10,5),"LEFT")
   draw.text((L/2+10,5),"RIGHT")
   del draw
#   tailleEntree[0]=L;
#   tailleEntree[1]=H;
#    n=input("Voulez-vous travailler en centrage \npar defaut tapez 1\n par vos
valeurs tapez 2 ? [1;2]");
   n=1; # travailler uniquement par defaut
   if ((n=="1") | (n==1)) :
      print ("Pas de centrage.")
      ptG.append(L/4);
      ptG.append(H/2);
      ptD.append(3*L/4);
      ptD.append(H/2);
   else : # à reimplementer
      # coordonnees d'un point commun si la source est mal centree
      ptG.append(752);#"valable uniquement pour test fichier nea3d-1186-1187"
      ptG.append(183);
      ptD.append(2300);
      ptD.append(186);



   print;
```

```
   # preparation du traitement
   # tailleSortie[1]=tailleEntree[1];
   # Les fichiers images G et D
   if ( (L-ptD[0]) < ((L/2)-ptG[0]) ) :
      Ld = (L-ptD[0])
   else : Ld = ((L/2)-ptG[0]);
   if ( (ptG[0]) < (ptD[0]-(L/2)) ) :
      Lg = (ptG[0])
   else : Lg = (ptD[0]-(L/2));
   bx1=ptG[0]-Lg
   bx2=ptD[0]-Lg
   print "Lg=",Lg," ;  Ld=",Ld;
   print "bx1=",bx1," ;  bx2=",bx2;
   print;
   if ( (H-ptD[1]) < (H-ptG[1]) ) :
      Hd =  (H-ptD[1])
   else : Hd = (H-ptG[1]);
   if ( (ptG[1]) < (ptD[1]) ) :
      Hg =  (ptG[1])
   else : Hg= (ptD[1]);
   by1=ptG[1]-Hg
   by2=ptD[1]-Hg
   print "Hg=",Hg," ;  Hd=",Hd;
   print "by1=",by1," ;  by2=",by2;
   print
   tailleSortie[0]=Lg+Ld;
   tailleSortie[1]=Hd+Hg;
   print "entree=[",L,",",H, "] ; sortie=",tailleSortie;
   box=(bx1,by1,bx1+tailleSortie[0],by1+tailleSortie[1]);
   left=im.crop(box);
   box=(bx2,by2,bx2+tailleSortie[0],by2+tailleSortie[1]);
   right=im.crop(box);
   # Traitement
   rL, gL, bL = left.split();
   rR, gR, bR = right.split();
   sortieA = Image.merge("RGB", (rR, gL, bL));
   sortieB = Image.merge("RGB", (rL, gR, bR));
   draw = ImageDraw.Draw(sortieA)
   draw.text((10,15),"sortie A")
   del draw
   draw = ImageDraw.Draw(sortieB)
   draw.text((10,15),"sortie B")
   del draw
   sortieA.show();
   sortieB.show();
   sortieA.save("A_"+nomFichier);
   sortieB.save("B_"+nomFichier);
   print "Traitement avec succes !"
print "Au revoir !"
```

[i] Vous pouvez voir sur ce lien la présentation de cette clé : https://wiki.inria.fr/sciencinfolycee/ClefISN

[ii] Cette première partie est basée sur le livre de Gilles DOWEK et al. intitulé « Informatique et Sciences du Numérique » code article : G13543 ; ISBN : 978-2-212-13543-5 édition Eyrolles.