Début méthodologie caméras

Programmation

Le programme *Webcaming* a pour fonction d'ordonner la prise de photos par les caméras D-Link en format JPG ainsi que l'analyse des ces photos par le programme *Readimg*. La prise d'image est effectuée à toutes les 12h et débutée de manière telle quelle soit à 7h et 19h. Ceci est contrôlé par le paramètre *dt* dans *Webcaming*, auquel on a attribué la valeur de 43200 secondes.

Le programme *Readimg* effectue l'analyse des pixels des photos JPG et classe les pixels selon leur appartenance aux catégories: feuilles, tiges et autres. La programmation pour la catégorie rouge à également été faite, pour pouvoir analyser la surface de rouge sur la variété de laitue rouge, mais pas utilisée (compte tenu que la laitue n'ait pas eu le temps de rougir). *Readimg* attribue une valeur de gris de 1 à 3(allant de blanc à noir) au pixels selon leur catégorie (feuille= 1 , tige= 2, autre= 3). Une image PGM est créé avec ces valeurs de gris. La superficie de pixels ayant la valeur correspondant à feuille est inscrite dans un fichier texte. Cette superficie est calculée en faisant le produit de la surface d'un pixel en mètre et le nombre de pixels de valeur feuille.

Voici la présentation du programme *Webcaming* d'origine, écrite par Martin Aubé

```
#!/bin/bash
# script pour prendre une image de la ip webcam
# supporte la ip webcam linksys et la d-link mais il faut definir le model
# sur la ligne ci-dessous
# valeurs cammodel="dlink" ou cammodel="linksys"
#
#    Copyright (C) 2010  Martin Aube
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
#    Contact: martin.aube@cegepsherbrooke.qc.ca
#
#
```

Le programme contient plusieurs boucles traitant des caméras d'un même groupe. Les caméras 100 et 101 forment un groupe qui correspond aux caméras des basilics sous les HPS. Les 102 et 103 aux caméras des laitues sous les HPS . Les 104 et 105 aux caméras des basilics sous la lampe solaire. Les 106 et 107 aux caméras des laitues sous la lampe solaire .

Voici la boucle de 100 et 101, précédée de quelques déterminations de variables:

```
#définition de la variable date pour utilisation dans le nom des fichiers image
date=`date +"%H"`

# dt est le délai entre les prises d'images en secondes
let dt=43200
# programme qui fonctionne avec une camera de marque dlink
let n=1
while [ $n -gt 0 ]
#on nomme les fichiers image par la date et le temps auxquels ils ont été pris
do noname=`date +%Y-%m-%d_%H:%M:%S`
# seulement les caméras dont l'IP est 100 ou 101 sont concernées par cette boucle
        liste1="100 101 "
        #pour mieux identifier le processus de quelle caméra est en cours dans le terminal
           echo "============================================="
           echo $ncam
           echo ""
#les fichiers txt contenant les bonnes valeurs de référence pour 100 et 101 sont copiés dans les fichiers
txt utilisés pour l'analyse
        cp -f feuille1-1.txt feuille.txt
        cp -f tige1-1.txt tige.txt
        cp -f autre1-1.txt autre.txt
        cp -f rouge1-1.txt rouge.txt
        for ncam in $liste1
# la fonction wget du lien associé à la caméra d'un IP $ncam cause la prise d'image par la caméra
        do /usr/bin/wget http://sand:memo123@192.168.0.$ncam/IMAGE.JPG
#envoi de l'image pour l'analyse
           /bin/mv -f IMAGE.JPG $ncam.jpg
           /usr/bin/convert -compress none $ncam.jpg image.ppm
           # on s'assure que le dossier existe
           /bin/mkdir /home/lapinfougace/fichier-Antoine/192.168.0.$ncam
           # on bouge l'image dans son dossier
           /bin/mv -f $ncam.jpg /home/lapinfougace/fichier-Antoine/192.168.0.$ncam/$ncam-$noname.jpg
           # readimg doit seulement imprimer les valeurs inscrites après le echo suivant dans le fichier
txt evolution_s_feuille1_
           # ces valeurs sont obtenues par la fonction read
           /home/lapinfougace/fichier-Antoine/readimg > npix.tmp
           read surface npix ntige nautre nrouge < npix.tmp
           /bin/mv -f feuille.pgm /home/lapinfougace/fichier-Antoine/192.168.0.$ncam/$ncam-$noname"_1.pgm"
           echo $noname $ncam $surface $npix $ntige $nautre $nrouge >> /home/lapinfougace/fichier-Antoine/
evolution_s_feuille1_$ncam.txt
        done
```

Les boucles restantes suivent.

Ensuite, il y a mise en veille du programme pour un certain temps, correspondant au délai de la prise de photo. Cette partie est la “fin” du programme.

```
        sleep $dt
done
#fi
```

Voici des captures d’écran du programme *Readimg* :

```
c programme pour lire et calculer a partir des couleurs d'une image ppm
c
c Declaration des variables
       integer i,j,nx,ny,k,n,nf,na,nt,nr,ndf,nda,ndt,ndr,malf,totf
       integer rtot, gtot, btot, intensity, seuil_nuit
       integer imageo(640,480)
       real image(640,480,3),scale,ratior,ratiob,ratioj,seuil,som
       real feuille(200,3),rouge(200,3)
       real tige(200,3)
       real autre(200,3),d,dmin
c initialisation des variables
       n=0
c scale est la taille d'un pixel en m
       scale=0.001
c programme principal
       open(unit=1,file='image.ppm',status='unknown')
        read(1,*)
        read(1,*) nx,ny
        read(1,*)
        read(1,*) (((image(i,j,k),k=1,3),i=1,640),j=1,480)
       close(unit=1)
c lecture automatique des pixels
c (détection feuille)
       open(unit=2,file='feuille.txt',status='unknown')
       read(2,*) nf
       do i=1,nf
         read(2,*) feuille(i,1),feuille(i,2),feuille(i,3)
       enddo
      close(unit=2)
c (détection tige)
      open(unit=2,file='tige.txt',status='unknown')
       read(2,*) nt
       do i=1,nt
         read(2,*) tige(i,1),tige(i,2),tige(i,3)
       enddo
      close(unit=2)

c (détection autre)
      open(unit=2,file='autre.txt',status='unknown')
       read(2,*) na
       do i=1,na
         read(2,*) autre(i,1),autre(i,2),autre(i,3)
       enddo
      close(unit=2)

c (détection rouge)
      open(unit=2,file='rouge.txt',status='unknown')
       read(2,*) nr
       do i=1,nr
         read(2,*) rouge(i,1),rouge(i,2),rouge(i,3)
       enddo
      close(unit=2)
```

Webcamimg est le responsable de la conversion des images de format JPG en PPM pour en permettre l'analyse par *Readimg*.

```
      do j=1,480
         do i=1,640
            dmin=10000000.
c imageo=1 => feuille imageo=2 => tige imageo=3 => autre  rouge imageo=4
            do n=1,nf
              d=sqrt((image(i,j,1)-feuille(n,1))**2.+(image(i,j,2)-
     +        feuille(n,2))**2.+(image(i,j,3)-feuille(n,3))**2.)
              if (d.lt.dmin) then
                 dmin=d
                 imageo(i,j)=1
              endif
            enddo
            do n=1,nt
              d=sqrt((image(i,j,1)-tige(n,1))**2.+(image(i,j,2)-
     +        tige(n,2))**2.+(image(i,j,3)-tige(n,3))**2.)
              if (d.lt.dmin) then
                 dmin=d
                 if(imageo(i,j).ne.0) then
                     malf = malf + 1
                 endif
                 totf = totf + 1
                 imageo(i,j)=2
              endif
            enddo
c            print *, malf / totf
            do n=1,na
              d=sqrt((image(i,j,1)-autre(n,1))**2.+(image(i,j,2)-
     +        autre(n,2))**2.+(image(i,j,3)-autre(n,3))**2.)
              if (d.lt.dmin) then
                 dmin=d
                 imageo(i,j)=3
              endif
            enddo
            do n=1,nr
              d=sqrt((image(i,j,1)-rouge(n,1))**2.+(image(i,j,2)-
     +        rouge(n,2))**2.+(image(i,j,3)-rouge(n,3))**2.)
              if (d.lt.dmin) then
                 dmin=d
                 imageo(i,j)=4
              endif
            enddo


            write(2,*) imageo(i,j)
            if (imageo(i,j).eq.1) ndf=ndf+1
            if (imageo(i,j).eq.2) ndt=ndt+1
            if (imageo(i,j).eq.3) nda=nda+1
            if (imageo(i,j).eq.4) ndr=ndr+1
            write(1,*) imageo(i,j)



         enddo
      enddo
      close(unit=1)
c sur l'image pgm gris fonce=feuille, gris pale = tige, blanc=autre
      print*,scale*scale*real(ndf),ndf,ndt,nda,ndr
c le premier nombre est la surface de feuille en metre carre et il
c faut toutefois definir scale comme la taille en m d'un pixel
c ndf est le nombre de pixel de feuille
c ndt est le nombre de pixel de tige
c nda est le nombre de pixel autres
c ndr est le nombre de pixel rouges

 100  format('P2')

      stop
      end
```

C'est l'attribution de la valeur d'*imageo* qui détermine la couleur d'un pixel sur l'image PGM. Le code concernant *malf* et *totf* n'est pas terminé et c'est pourquoi la commande *print * malf*

/ *totf* est en commentaire. Il est une ébauche dans un processus de vérification de la validité du classement des pixels dans la catégorie feuille.