Aix*Marseille université

Faculté des Sciences

Licence 3 SPI

Practical works

Introduction to Raspberry Pi

Examples of Python programmes

Julien Marot

1 Image processing algorithms under Python

1.1 Algorithm dedicated to USB camera: mathematical morphology

Algorithm 1

```
myCamera= Camera...
(prop_set={'width':320,'height':240})
myDisplay=Display(resolution=(320,240))
frame=myCamera.getImage()
frame.save("camera-output.jpg")
binarized = frame.binarize(0.5)
dilated=binarized.dilate(0.5)
closed=dilated.erode(0.5)
```

1.2 Algorithm dedicated to Pi camera: mathematical morphology

Algorithm 2

```
camera = picamera.PiCamera()
myDisplay=Display(resolution=(320,240))
frame = camera.capture('image.jpg')
frame.save("camera-output.jpg")
binarized = frame.binarize(0.5)
dilated=binarized.dilate(0.5)
closed=dilated.erode(0.5)
```

1.3 Algorithm dedicated to Pi camera: face detection

Algorithm 3 Image acquisition

```
camera = picamera.PiCamera()
frame = camera.capture('image.jpg')
sleep(.1)
```

Algorithm 4 Face detection

```
frame = camera.capture('image.jpg')
faces = frame.findHaarFeatures('face')
  if faces:
      for face in faces:
         print "Face at: " + ...
         str(face.coordinates())
   else:
      print "No faces detected."
```

Algorithm 5 Image display

```
myDisplay = Display...
(resolution=(320, 240))
frame.save(myDisplay)
while not myDisplay.isDone():
   sleep(0.1)
```

Algorithm 6 Whole algorithm Pi camera and face detection

```
##import picamera
import cv2

## Take a photograpy, save as image.jpg and shut down the camera
camera = picamera.PiCamera()   ## Declare the camera
camera.resolution = (1024,768) ## Choose the resolution
camera.capture('image.jpg')    ## Capture a snapshot
camera.close()                 ## Close the camera

## Load the classifier for face detection
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
## The xml file must be present in the current folder

## Read the image
image = cv2.imread('image.jpg')

## Detect the faces
faces = faceCascade.detectMultiScale(image, 1.3, 5)

## Draw the rectangles on the faces:
for(x, y, w, h) in faces:
    cv2.rectangle(image, (x,y), (x+w, y+h),(0, 255, 0), 2)

## Display the image with detected faces
namewindow = "Faces found"
cv2.imshow(namewindow,image)            ## Draw the image
##cv2.resizeWindow(namewindow,800,600)  ## Possibly resize the window
cv2.waitKey(0)                          ## Press a key ...
cv2.destroyWindow(namewindow)           ## ... to close the window
```

2 Data transfer under Python

2.1 Data sending

Algorithm 7 Send a file

```
import socket               # Import socket module

s = socket.socket()         # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345                 # Reserve a port for your service.

s.connect(("192.168.1.86", port))

filetosend = raw_input()
f = open(filetosend,'rb')
l=f.read(1024)
while (l):
    print 'Sending...'
    s.send(l)
    l = f.read(1024)
f.close()
print "Done Sending"
s.shutdown(socket.SHUT_WR)
print s.recv(1024)
s.close                     # Close the socket when done
```

2.2 Data reception

Algorithm 8 receive a file

```
import socket               # Import socket module

s = socket.socket()         # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345                # Reserve a port for your service.
s.bind((host, port))        # Bind to the port
f = open('torecv','wb')
s.listen(5)                 # Now wait for client connection.
while True:
    c, addr = s.accept()     # Establish connection with client.
    print 'Got connection from', addr
    print "Receiving..."
    l = c.recv(1024)
    while (l):
        print "Receiving..."
        f.write(l)
        l = c.recv(1024)
    f.close()
    print "Done Receiving"
    c.send('Thank you for connecting')
    c.close()                # Close the connection
```