eBook Gratuit

APPRENEZ computer-vision

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#computer-vision

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: computer-vision

It is an unofficial and free computer-vision ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official computer-vision.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec la vision par ordinateur

Remarques

Le traitement d'image numérique et la vision par ordinateur sont un domaine intéressant car ils se situent magnifiquement entre les mathématiques et l'informatique. Par conséquent, il est très utile de comprendre les principes fondamentaux et de les appliquer en utilisant la programmation pour comprendre le sujet.

Les images numériques sont la discrétisation de signaux à 2 ou 3 dimensions. En d'autres termes, les images numériques sont des ensembles de pixels ou de voxels échantillonnés de domaines continus.

$$f : R^2 \supset \Omega \to R$$

Où f est une image numérique sur Ω: Domaine d'image rectangulaire

Par souci de simplicité, nous ne discuterons que des images numériques bidimensionnelles, comme celles de nos avatars StackOverflow.

À propos des pixels: Une note rapide sur les valeurs des pixels avant de commencer à discuter des types d'images. En règle générale, les pixels commencent par la valeur 0, qui indique l'absence de lumière (noir), atteint 1, l'intensité maximale (par exemple, le blanc) et sont représentés par des entiers.

Images binaires: images en noir et blanc uniquement. Chaque pixel est 0 ou 1, chaque pixel peut être représenté par un bit. Ils ne sont pas très populaires, car ils sont généralement utilisés dans des applications scientifiques ou pour d'autres opérations de traitement d'images en tant que masque par exemple.

Un exemple d'image binaire. (Attention les valeurs de pixel de l'image de ce fichier ne sont pas

nécessairement binaires, ceci est pour démonstration, c'est aussi Lena, la star du monde du traitement d'image)

Images en niveaux de gris: grâce aux filtres en ligne, tout le monde connaît encore très bien ces images. Ces images sont généralement un octet par pixel, avec 0 étant noir et 255 étant blanc, tout entre les deux est un gris différent, étant donné que les humains ne peuvent distinguer que 40 nuances de gris, cette plage est suffisante pour de nombreuses applications (notez que les valeurs des pixels ici sont mappés de 0 à 1 aux valeurs d'octet 0 - 255)

Images couleur: Enfin, le type d'image numérique le plus courant, les images couleur. Nous devons mentionner le concept de chaînes ici. Les images numériques ont également des canaux. En réalité, les images binaires et en niveaux de gris décrites ci-dessus ont également des canaux. La description la plus courante serait le modèle RVB (Rouge-Vert-Bleu). Dans ce cas, l'image comporte 3 canaux (ne pas confondre avec les dimensions, ce sont toujours des images 2D) pour décrire la rougeur, le bleu et la verdure de l'image. Dans ce cas, chaque pixel est un triplet, une valeur comprise entre 0 et 255 (pas de rouge à plus rouge), 0 - 255 (pas de vert à plus vert), 0 - 255 (pas de bleu à plus de bleu). Pour ce modèle, le pixel {0,0,0} est noir, {255,255,255} est blanc, {255,0,0} est rouge, {255, 255, 0} est jaune. Cependant, la couleur est un sujet très vaste et vous pouvez vérifier les références pour plus d'informations.

Images hyper-spectrales:

Après avoir discuté des canaux, parler d'images hyper-spectrales est plus facile. Ces images peuvent comporter des centaines de canaux et sont couramment utilisées en microscopie, en imagerie par satellite, etc.

Lectures

1. Échantillonnage du signal: https://en.wikipedia.org/wiki/Sampling_(signal_processing)
2. Bible de traitement d'images numériques: RC Gonzalez, RE Woods: Traitement d'images numériques. Troisième édition, Pearson Prentice Hall, Upper Saddle River, 2008.
3. Revue de vision par ordinateur (jusqu'à l'apprentissage en profondeur): R. Szeliski: Vision par ordinateur: algorithmes et applications. Springer, New York, 2010.
4. Pour comprendre les images binaires, en niveaux de gris et en couleur: https://en.wikipedia.org/wiki/Grayscale

Examples

Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de la vision par ordinateur.

Pour cette série de vision par ordinateur et les suivantes, j'utiliserai Python 2 comme langage de programmation. Python est un choix courant pour la communauté scientifique, il est gratuit, il a beaucoup de bibliothèques gratuites et open-source, et si vous êtes novice en programmation, il est l'un des plus simples à apprendre et à commencer à programmer.

Maintenant installé, si vous utilisez Linux, vous avez probablement déjà python, ouvrez simplement le terminal et tapez "python" pour vérifier si tout fonctionne déjà. D'autres peuvent consulter ce lien et télécharger python 2.7.

Deuxièmement, nous devrons installer les bibliothèques que nous allons utiliser dans le code source. Maintenant, une note ici, cette configuration est conçue pour cet exemple, dans les étapes ultérieures, nous allons ajouter différentes bibliothèques, et, bien sûr, les différentes applications de vision par ordinateur peuvent nécessiter des bibliothèques spécifiques telles que OpenCV. Une seule bibliothèque sera installée sur notre système pour exécuter le code. Lors de l'utilisation de python, j'installe généralement des dépendances utilisant 'pip'. C'est un outil simple pour installer les modules python, vous pouvez aussi le vérifier via ce lien

Maintenant, nous sommes prêts à installer la bibliothèque dont nous avons besoin, PyPNG. Si vous utilisez pip, il vous suffit de

```
pip installer PyPNG
```

dans le terminal si vous utilisez Linux / Mac, en ligne de commande si vous utilisez Windows.

De plus, pour ces exercices, vous devez obtenir des images qui peuvent être trouvées dans le lien github avec le code source et les carnets d'ipython.

https://github.com/Skorkmaz88/compvis101

Maintenant, nous devrions être bons pour faire de l'exercice

Exemples

Il s'agit d'une série d'exercices Python de traitement d'images et de vision par ordinateur très simples, conçus pour introduire ces sujets avec peu de pratique. Je suis désolé par avance pour toute erreur de recrue, elle est encore en développement. Dans cette série, je limiterai les images numériques que nous pouvons utiliser aux fichiers PNG dans un souci de simplicité, dont je parlerai également de la compression des images.

Veuillez cloner le dépôt si vous ne l'avez pas déjà fait, ou simplement le télécharger ici via Github:

```
git clone https://github.com/Skorkmaz88/compvis101
```

Vous pouvez utiliser deux fichiers, tutorial0.py et readingImages.ipynb. Le second est un cahier ipython si vous préférez l'utiliser. Mais deux fichiers font la même chose.

Code a des explications dans les commentaires, je suis

```
 # libs
import png

# We create a greyscale image as described in our text.
# To do that simply, we create a 2D array in python.
# x and y, x being horizontal and y being vertical directions.

x  = []
y = []
# Play around with these pixels values to get different grayscale images, they shoud be
# in range of 0 - 255.
white = 255
gray = 128
black = 0
width  = 100
height = 300

# Add 100 x 100 rectangle as just white(255) valued pixels
for i in range(0, 100):
    for j in range(0,100):
        y.append(white); # Pixel (i,j) is being set to a value, rest is coding trick to nest
two lists
    x.append(y)
    y = []

# Add 100 x 100 rectangle as just mid-gray(128) valued pixels
for i in range(0, 100):
    for j in range(0,100):
        y.append(gray);
    x.append(y)
    y = []

# Add 100 x 100 rectangle as just black(0) valued pixels
```

```
for i in range(0, 100):
    for j in range(0,100):
        y.append(black);
    x.append(y)
    y = []

# output image file
f = open('out.png', 'wb')
w = png.Writer(width, height , greyscale=True, bitdepth=8)
w.write(f, x)
f.close()
# If everything went well, you should have 3 vertically aligned rectangles white, gray and
black
# Check your working folder

# PART 2
# Read a grayscale image and convert it to binary

# This time we will binarize a grayscale image, to do that we will read pixels and according
to threshold we set
# we will decide if that pixel should be white or black

# This file is originally 8 bit png image, can be found in github repository, you should use
only this type of
# images if you want to change the image.
f = open('./img/lenaG.png', 'r')

r=png.Reader(file=f)
# You will the details about the image, for now pay attention to size and bitdepth only.
img = r.read()

width = img[0]
height = img[1]
# Threshold value for binarizing images,
threshold = 128
print "Input image size is: "+ str(width)+ " pixels as  width, " + str(height) + " pixels as
height"

f_out = open('lenaBinary.png', 'wb')
w = png.Writer(width, height , greyscale=True, bitdepth=1)

pixels = img[2]

x = []
y = []

# Let's traverse the Lena image
for row in pixels:
    for pixel in row:
        p_value =  pixel
        # Now here we binarize image in pixel level
        if p_value > threshold:
            p_value = 1
        else:
            p_value = 0

        y.append(p_value);
    x.append(y)
    y = []

w.write(f_out, x)
```

```
f_out.close()
```

Si tout a bien fonctionné, félicitations! Vous avez créé une image à partir de zéro et effectué la première transformation du niveau de pixel sur une image existante. Vérifiez votre dossier de travail pour voir les nouvelles images

Lire Démarrer avec la vision par ordinateur en ligne: https://riptutorial.com/fr/computer-vision/topic/5710/demarrer-avec-la-vision-par-ordinateur

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec la vision par ordinateur	Community, Semih Korkmaz