PROJET
Imagerie Numérique

Nguyen Nathalie

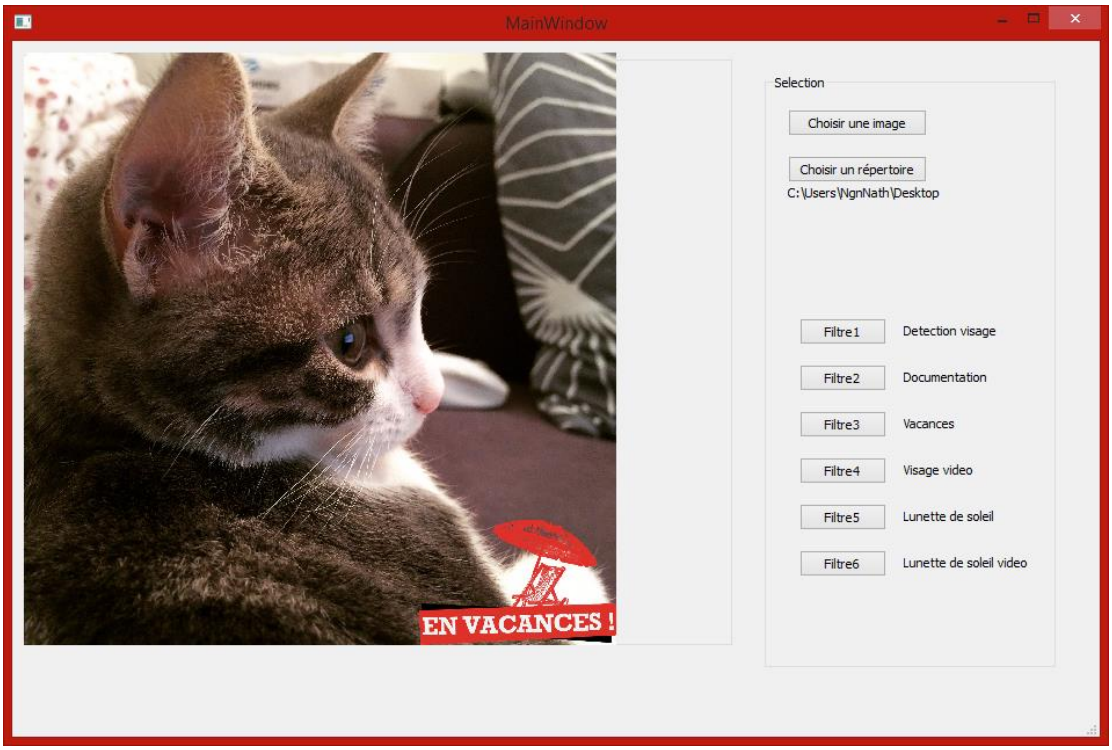
4A-CFA

ESIEA

Juin 2017

Sommaire

1. Description de l’application

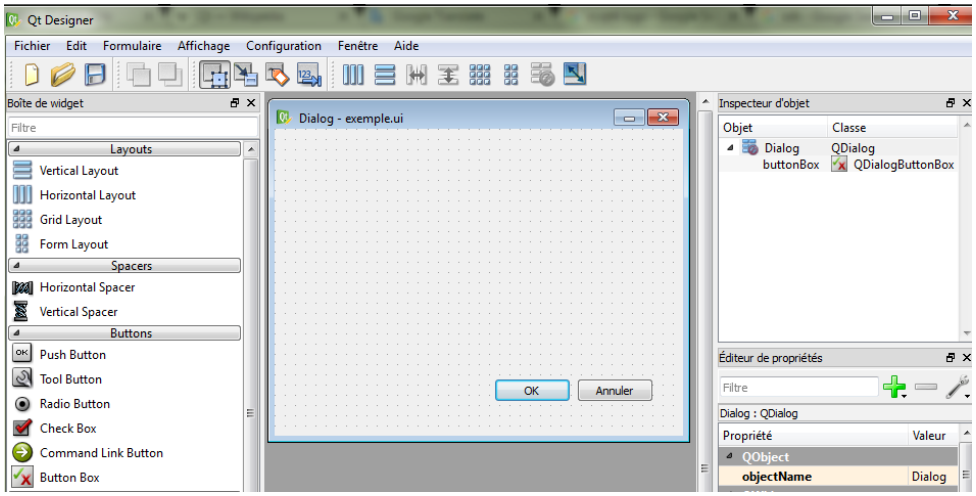


Ce projet est une application permettant de mettre des filtres soit sur une image ou via la caméra. Cette application vous donne la possibilité de choisir parmi les filtres ci-dessus. Les filtres s’activent via les boutons à droite de la fenêtre. Il est possible d’enregistrer la photo modifiée sur le chemin que vous aurez préalablement indiquer.

2. Interface

La bibliothèque choisie pour implémenter l'interface est PyQt4, un ensemble qui relie le langage Python avec Qt, un SDK multi-plateforme pour conceptualiser, développer et maintenir les interfaces graphiques. L'installation de PyQt peut s'effectuer de plusieurs manières, mais nous avons choisi d'utiliser le package Python (fichier téléchargé d'extension .whl) avec le gestion d'installation *pip install*.

Pour la conception, j'ai d'abord utilisé le logiciel Qt Designer qui me permet de faire la conception de notre IHM.



Le projet étant sauvegardé sous le format .ui, ce fichier décrit l'interface avec la syntaxe xml. Il est possible de générer le code python:

```
pyuic4 -o imagerie_gui.py -x imagerie_gui.ui
```

Nous avons notre interface statique, il ne reste plus qu'à interfacer avec les futurs filtres.

```
self.bouton.clicked.connect(self.fonction_filtre)
```

Pour ouvrir un fichier, nous utilisons QtGui.QFileDialog

```
filepath = QFileDialog.getOpenFileName(self.centralwidget, 'Open File', '/')
```

La mise à jour s'effectue :

```
image = QImage(filepath)
self.pic.setPixmap(QtGui.QPixmap(QPixmap.fromImage(image)))
```

Pour exécuter un filtre :

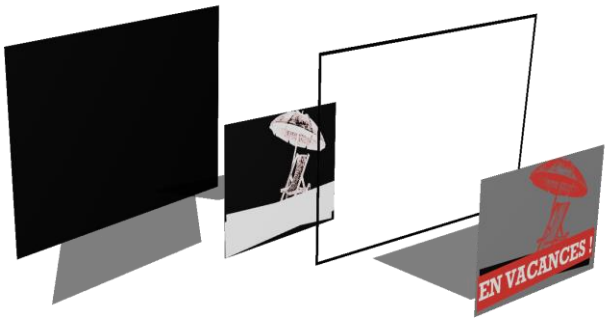
```
os.system("python code_python/filtre_detection_visage-video/face_img.py")
```

Si on choisit un fichier en particulier, et que nous souhaitons récupérer le résultat :

```
cmd="python code_python/filtre_doc/traitement_document.py
"+self.current_image
result = subprocess.check_output(cmd, shell=True)
```

3. Les filtres

a. Filtre sur une image



- **Création des fonds noirs et transparents**

On crée un fond noir et d'un fond transparent pour que les filtres correspondent au format de l'image que l'on souhaite modifier.

```
fond_noir = Image.new('RGB',(width,height),(0,0,0))
fond_noir.save("fond_noir.png", "PNG")
fond_transparent = Image.new('RGBA', (width,height), (0,0,0,0))
fond_transparent.save("fond_transparent.png", "PNG")
```

Par la suite on remplace une partie de l'image par notre motif que l'on veut faire apparaître sur la photo.

```
fond_transparent[x:x+largeur;y:y+hauteur]=motif[0:largeur;0:hauteur]
```

- **Redimension de l'image**

Pour redimensionner l'image on donne tout d'abord la largeur souhaitée puis via cette mesure on calcule le ratio (wpercent) pour avoir la nouvelle hauteur (hsize).
Par la suite on utilise la fonction resize avec les dimension et le rééchantillonnage PIL.Image.ANTIALIAS permettant de définir la qualité de l'image.

```
basewidth = x_lunette_fin - x_lunette_debut
img_to_resize = Image.open(nom_filtre+'.png')
wpercent = (basewidth / float(img_to_resize.size[0]))
hsize = int((float(img_to_resize.size[1]) * float(wpercent)))
img_resized = img_to_resize.resize((basewidth, hsize), PIL.Image.ANTIALIAS)

img_resized.save('resized_'+nom_filtre+'.png')
img_to_resize = Image.open(nom_filtre+'_fond.png')
img_resized = img_to_resize.resize((basewidth, hsize), PIL.Image.ANTIALIAS)
```

```
img_resized.save('resized_'+nom_filtre+'_fond.png')
```

- **Superposition de l'image**

On convertie l'image de uint8 en float, puis on normalise le filtre pour régler l'intensité du filtre transparent.
On multiplie les deux matrices filtre et filtre_fond pour avoir une matrice dans laquelle on garde les éléments que l'on veut.
Pour assembler l'image du filtre avec l'image modifié on additionne les matrices via la fonction cv2.add().

```
filtre = filtre.astype(float)
image_modif = image_modif.astype(float)
filtre_fond = filtre_fond.astype(float)/255
filtre = cv2.multiply(filtre_fond, filtre)
image_modif = cv2.multiply(1.0 - filtre_fond, image_modif)
outImage = cv2.add(filtre, image_modif)
```

b. Détection du visage et des yeux

On charge la bibliothèque de détection des visages et des yeux.

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
```

On convertit l'image via la fonction cv2.cvtColor() avec le paramètre image et le paramètre "cv2.COLOR_BGR2GRAY" la couleur à garder.

```
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Cette section de code permet de détecter le premier coin du visage par les coordonnées x et y et w correspond à la largeur et h la hauteur du visage.
On extrait de l'image en gris et en couleur le visage pour mettre dans l'image roi_gray et roi_color.

```
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
```

Ce code va détecter les yeux du visage.
Cette section de code permet de détecter le premier coin de l'œil par les coordonnées ex et ey et ew correspond à la largeur et eh la hauteur de l'œil.

```
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
       cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
```

Ces sections de code sont utilisées pour les filtre “troll” et “lunette”.

c. Filtre sur vidéo

Pour utiliser le flux vidéo de la webcam, on utilise la fonction VideoCapture.
En utilisant la fonction read() on arrive à extraire l’image et une variable booléenne qui est à True si l’image est bien lue.

```
video_capture = cv2.VideoCapture(0)
ret, frame = video_capture.read()
```

Pour appliquer les précédentes fonctionnalités, il suffit d’assembler les précédents code en prenant l’image de base la frame extraite de la capture vidéo.
Pour que le filtre s’applique en temps réel lors du lancement de la caméra, nous mettons ce code dans une boucle pour que cette fonction s’applique pour toutes les images transmises par la caméra.
Remarque : dû à une génération d’image trop importante et un traitement d’image important, il a fallu mettre un temps de 1 seconde pour que l’application ne plante pas.

d. Message sur une image

```
cv2.putText(img, 'Hello', (width/3,height/2), font, 1,(0,0,0),2,cv2.CV_AA)
```

4. Continuation possible

Nous avons plusieurs pistes pour améliorer le logiciel :

- Attribuer le nom de l’image à enregistrer
- Plusieurs choix de filtre
- Des options pour importer une image en tant que filtre
- Ouverture du dossier de destination
- Les codes erreurs lorsque le chemin d’enregistrement n’est pas indiqué ou le chemin de la photo à traiter.

Sources

https://pypi.python.org/simple/pyqt4-windows-whl/
https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/
http://axelbellec.fr/articles/2015/12/01/facial-recognition.html