RAPPORT PROJET

TRAITEMENT MULTIMEDIAS

SUJET

DECTECTION DE SIGNATURE

Auteurs :

SOKRI Yasser

OUBADDOU Abdellah

Encadrant :

Pr. N. ABOUTABIT

29 MAI 2022

Sommaire

I. GENERALITES

Aussi vielle qu'est le système administratif actuel, adopté par les entreprises et les institutions étatiques, la signature fait partie des plus anciens moyens d'identifier l'auteur d'un document ou d'une œuvre en général.

Unique et invariante dans le temps, elle peut être manuscrite(faite à la main) ou virtuelle(électronique) et s'appose sur tous types de supports.

Dans notre projet, nous nous sommes orientés vers les signatures contenues dans des documents administratifs.

Etant donné leur utilisation accrue pour authentifier les documents, il s'avère important d'optimiser les processus d'archivages des documents en s'assurant qu'ils sont tous conformes.

L'idée est alors de pouvoir automatiser la détection des signatures dans les documents administratifs après scanage préalable pour une reconnaissance probable plus tard en cas de besoin.

II. APPROCHE DE RESOLUTION

Une fois scanner, il faut alors définir un moyen numérique de traiter ces documents de manière automatisée. Une panoplie de solutions et d'approches sont envisageables avec leurs faiblesses et leurs forces.

Le langage de programmation python propose une librairie dédiée pour la vision par ordinateur (cv2) qui permet d'effectuer les traitements nécessaires et récurrents sur les images. C'est donc ce langage qui a été utilisé pour réaliser notre projet.

En effet il existe différents formats de documents administratifs dont le plus utilisé est le format A4(2480x3508 pixels).

Pour détecter les signatures dans notre document, il fallait déterminer un intervalle où les signatures s'existent, c-à-d, pratiquement, déterminer deux seuils, le premier seuil a pour objectif d'éliminer les petits éléments, notamment les lettres, et les chiffres, et le deuxième seuil pour éliminer les grandes parties comme les images et les logos. Nous détaillons cela dans les parties qui viennent.

Dans le processus qu'on a suivi, il y avait deux étapes majeurs le prétraitement et le traitement.

III. IMPLEMENTATION SOUS PYTHON

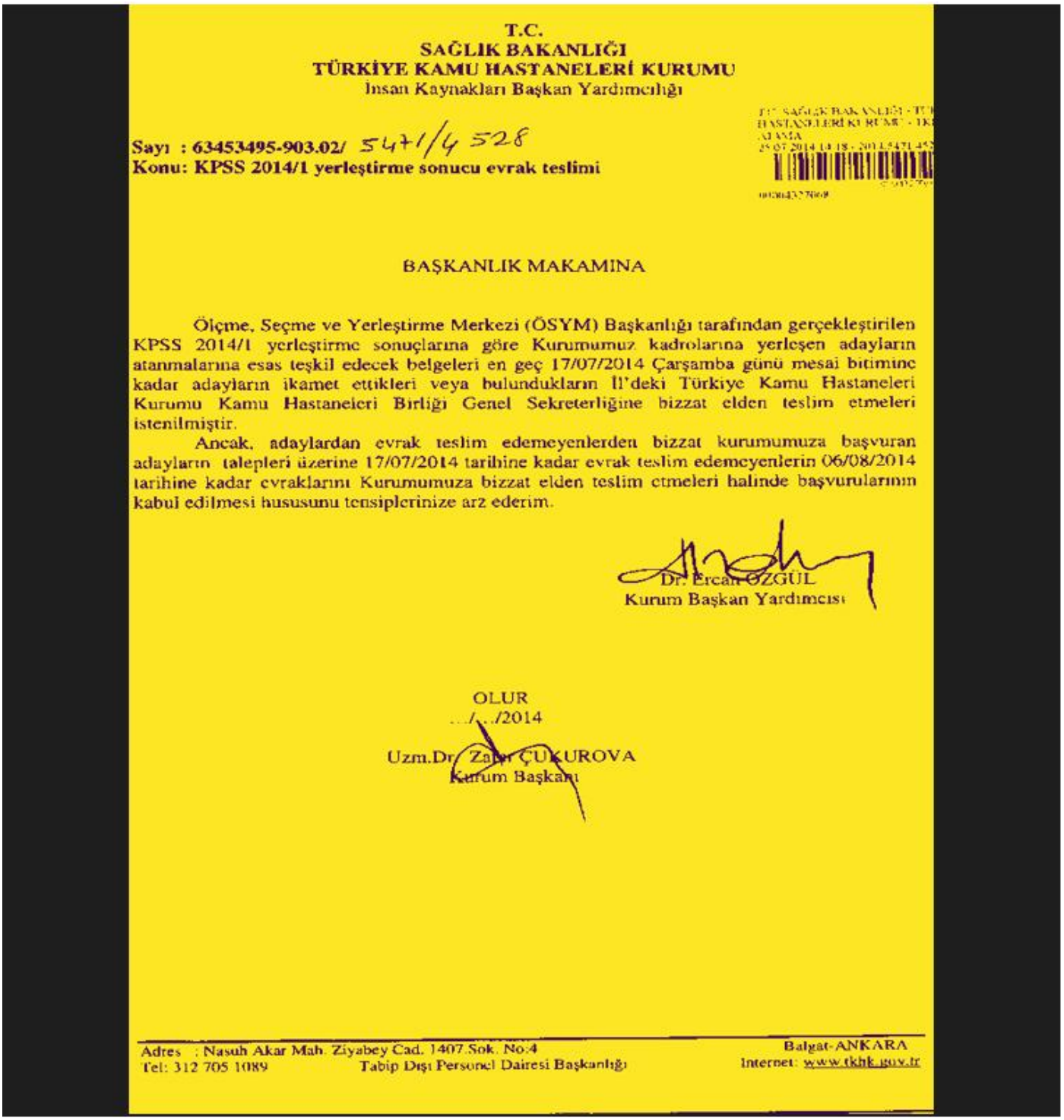
IDENTIFICATION DE L'INTERVALLE DE SIGNATURE

Pré-traitement

Conversion de l'image en une image en niveau de gris et binarisation de l'image

```
# lecture de l'image entrée
img = cv2.imread(filename, 0)
#conversion niveau de gris et binarisation
img=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

plt.imsave('image_binariser.png',img)
```



T.C.
SAĞLIK BAKANLIĞI
TÜRKİYE KAMU HASTANELERİ KURUMU
İnsan Kaynakları Başkan Yardımcılığı

Sayı : 63453495-903.02/ 5471/4 528
Konu: KPSS 2014/1 yerleştirme sonucu evrak teslimi

BAŞKANLIK MAKAMINA

Ölçme, Seçme ve Yerleştirme Merkezi (ÖSYM) Başkanlığı tarafından gerçekleştirilen KPSS 2014/1 yerleştirme sonuçlarına göre Kurumumuz kadrolarına yerleşen adayların atanmalarına esas teşkil edecek belgeleri en geç 17/07/2014 Çarşamba günü mesai bitimine kadar adayların ikamet ettikleri veya bulundukları İl'deki Türkiye Kamu Hastaneleri Kurumu Kamu Hastaneleri Birliği Genel Sekreterliğine bizzat elden teslim etmeleri istenilmiştir.

Ancak, adaylardan evrak teslim edemeyenlerden bizzat kurumumuza başvuran adayların talepleri üzerine 17/07/2014 tarihine kadar evrak teslim edemeyenlerin 06/08/2014 tarihine kadar evraklarını Kurumumuza bizzat elden teslim etmeleri halinde başvurularının kabul edilmesi hususunu tensiplerinize arz ederim.

Dr. Ercan ÖZGÜL
Kurum Başkan Yardımcısı

OLUR
.../.../2014

Uzm.Dr. Zafer ÇUKUROVA
Kurum Başkanı

Adres : Nasuh Akar Mah. Ziyabey Cad. 1407.Sok. No:4 Balgat-ANKARA
Tel: 312 705 1089 Tabip Dışı Personel Dairesi Başkanlığı Internet: www.tkhk.gov.tr

Traitement

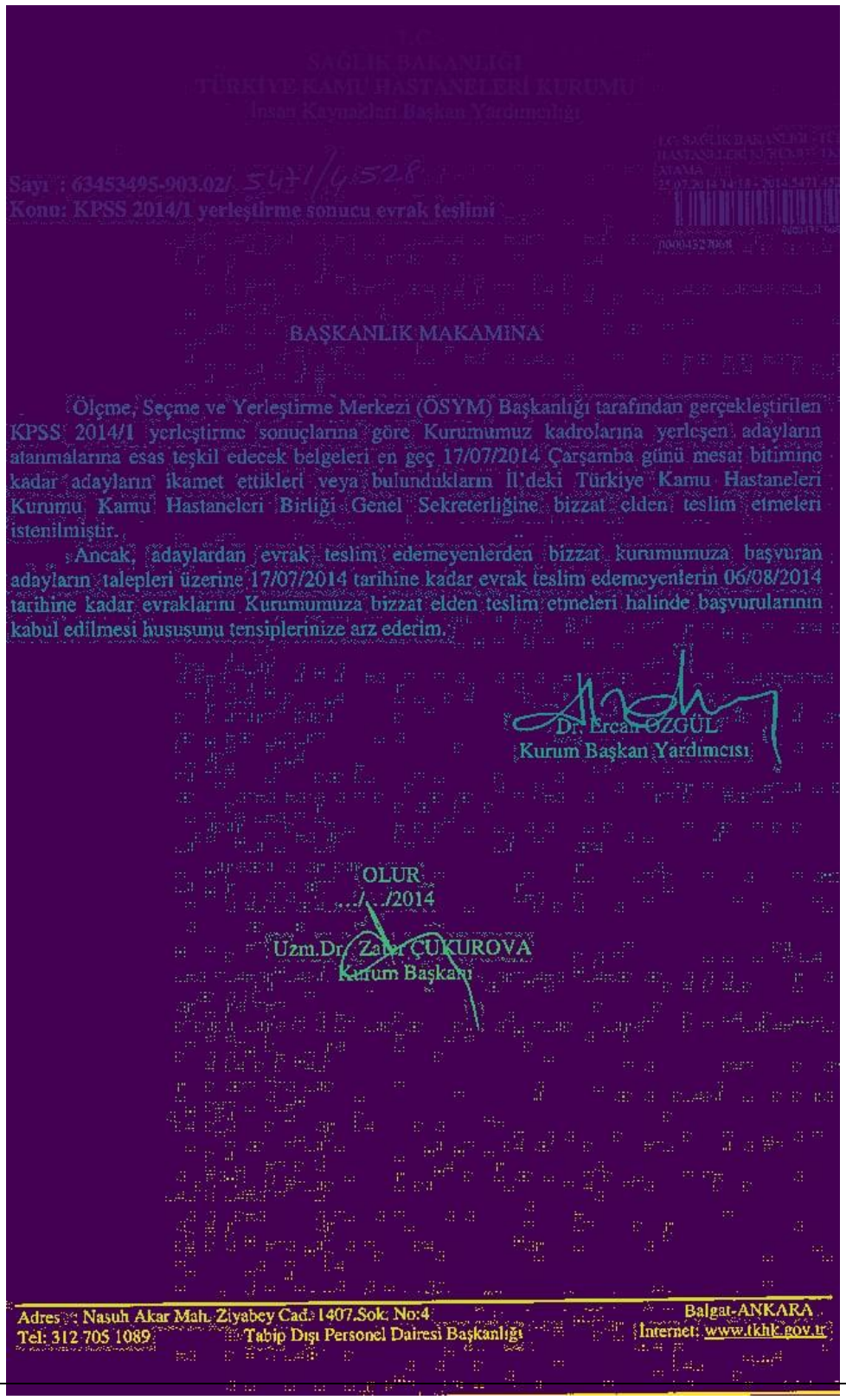
Analyser les composants connectés dans notre image

Pour ce fait, on a eu recours à la bibliothèque **« Scikit-Image »** prédéfinie sur python, dédié pour le traitement d'image

```
from skimage import measure, morphology
from skimage.color import label2rgb
from skimage.measure import regionprops
```

Maintenant nous définissons une condition qui va nous permettre de labeliser notre image.

```
blobs = img > img.mean()
blobs_labels = measure.label(blobs, background=1)
plt.imsave('image_labeliser.png',blobs_labels)
image_label_overlay = label2rgb(blobs_labels, image=img)
```



SAĞLIK BAKANLIĞI
TÜRKİYE KAMU HASTANELERİ KURUMU
İnsan Kaynakları Başkan Yardımcılığı

Sayı : 63453495-903.02/ 5471/4528
Konu: KPSS 2014/1 yerleştirme sonucu evrak teslimi

BAŞKANLIK MAKAMINA

Ölçme, Seçme ve Yerleştirme Merkezi (ÖSYM) Başkanlığı tarafından gerçekleştirilen KPSS 2014/1 yerleştirme sonuçlarına göre Kurumumuz kadrolarına yerleşen adayların atanmalarına esas teşkil edecek belgeleri en geç 17/07/2014 Çarşamba günü mesai bitimine kadar adayların ikamet ettikleri veya bulundukların İl'deki Türkiye Kamu Hastaneleri Kurumu Kamu Hastaneleri Birliği Genel Sekreterliğine bizzat elden teslim etmeleri istenilmiştir.

Ancak, adaylardan evrak teslim edemeyenlerden bizzat kurumumuza başvuran adayların talepleri üzerine 17/07/2014 tarihine kadar evrak teslim edemeyenlerin 06/08/2014 tarihine kadar evraklarını Kurumumuza bizzat elden teslim etmeleri halinde başvurularının kabul edilmesi hususunu tensiplerinize arz ederim.

Dr. Ercan ÖZGÜL
Kurum Başkan Yardımcısı

OLUR
../../2014

Uzm.Dr. Zafer ÇUKUROVA
Kurum Başkanı

Adres : Nasuh Akar Mah. Ziyabey Cad. 1407.Sok. No:4 Balgat-ANKARA
Tel: 312 705 1089 Tabip Dışı Personel Dairesi Başkanlığı Internet: www.tkhk.gov.tr

En arrivant à cette étape, nous devons éliminer les petits et les grands éléments de notre document.

Alors, pour cela, on détermine deux seuils. Après avoir effectué plusieurs tests on a pu extraire ces paramètres (**constante_1**, **constante_2, constante_3, constant_4** et amplifier) et par la suite déterminer ces deux formules, pour le but de définir les seuils, et finalement segmenter la signature.

Les paramètres constants

```
#Ces paramètres sont utilisées pour supprimer les pixels aberrants connectés de petites tailles
constante_1 = 84
constante_2 = 250
constante_3 = 100

#Ce paramètre est utilisé pour supprimer les pixels aberrants connectés de grandes tailles
constante_4 = 18
```

Les formules des seuils

```
moyenne = (air_totale/compteur)

# valeur seuil utilisée pour supprimer les pixels qui sont inférieurs à seuil_petit_element
# pour les documents numérisés au format A4
seuil_petit_element  = ((moyenne/constante_1)*constante_2)+constante_3

#  valeur seuil utilisée pour supprimer les pixels qui sont supérieurs à seuil_grand_element
# pour les documents numérisés au format A4
seuil_grand_element  = seuil_petit_element *constante_4
```

Jusqu'à présent on a que déterminé les éléments connectés. Alors, il faut maintenant éliminer les éléments indésirables. On commence par les petits éléments. Nous allons recourir au module de scikit-image **« morphology »** et plus précisément la fonction **« remove_small_objects() ».** Puis on cherche la taille des composants (nombre de pixels par composants). Ensuite, on détermine les composants qui sont supérieur au seuil (grands ou petits éléments), et on affecte les valeurs booléennes à notre image traitée dernièrement après avoir annuler les petits composants. Si un pixel appartient à une région où cette dernière contient un nombre d'éléments supérieurs au seuil, il aura une valeur **« true ».**

Et finalement on enregistre l'image finale

```python
# suppression des pixels connectés inférieurs à seuil_petit_element
pre_version = morphology.remove_small_objects(blobs_labels, seuil_petit_element )
# suppression des pixels connectés supérieurs au seuil seuil_grand_element
#pour se débarrasser des pixels connectés indésirables tels que les en-têtes de tableau, etc.
taille_composant = np.bincount(pre_version.ravel())
plus_petit_element = taille_composant > (seuil_grand_element)

plus_petit_element_masque = plus_petit_element[pre_version]

pre_version[plus_petit_element_masque] = 0
# enregistrement de la pre-version qui est l'image étiquetée avec des couleurs
plt.imsave('pre_version.png', pre_version)
```

Définition de la fonction d'extraction

```
def extractor(filename):

    #Ces paramètres sont utilisées pour supprimer les pixels aberrants connectés de petites tailles
    constante_1 = 84
    constante_2 = 250
    constante_3 = 100

    #Ce paramètre est utilisé pour supprimer les pixels aberrants connectés de grandes tailles
    constante_4 = 18

    # lecture de l'image entrée
    img = cv2.imread(filename, 0)
    #conversion niveau de gris et binarisation
    img=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

    plt.imsave('image_binariser.png',img)

    # analyse de composants connectés par le framework scikit-learn
    blobs = img > img.mean()
    blobs_labels = measure.label(blobs, background=1)
    plt.imsave('image_labeliser.png',blobs_labels)
    image_label_overlay = label2rgb(blobs_labels, image=img)


    plus_grand_composant = 0
    air_totale = 0
    compteur = 0
    moyenne = 0.0
    for region in regionprops(blobs_labels):
        if (region.area > 10):
            air_totale = air_totale + region.area
            compteur = compteur + 1
        if (region.area >= 250):
            if (region.area > plus_grand_composant):
                plus_grand_composant = region.area

moyenne = (air_totale/compteur)

# valeur seuil utilisée pour supprimer les pixels qui sont inférieurs à seuil_petit_element
# pour les documents numérisés au format A4
seuil_petit_element  = ((moyenne/constante_1)*constante_2)+constante_3

#  valeur seuil utilisée pour supprimer les pixels qui sont supérieurs à seuil_grand_element
#  pour les documents numérisés au format A4
seuil_grand_element  = seuil_petit_element *constante_4


# suppression des pixels connectés inférieurs à seuil_petit_element
pre_version = morphology.remove_small_objects(blobs_labels, seuil_petit_element )
# suppression des pixels connectés supérieurs au seuil seuil_grand_element
#pour se débarrasser des pixels connectés indésirables tels que les en-têtes de tableau, etc.
taille_composant = np.bincount(pre_version.ravel())
plus_petit_element = taille_composant > (seuil_grand_element)

plus_petit_element_masque = plus_petit_element[pre_version]

pre_version[plus_petit_element_masque] = 0
# enregistrement de la pre-version qui est l'image étiquetée avec des couleurs
plt.imsave('pre_version.png', pre_version)

# lecture de la pre-version
img = cv2.imread('pre_version.png', 0)
```

```
img = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
# sauvegarde
cv2.imwrite("output.png", img)
```

IV. CONCLUSION

Ce projet est le fruit d'un travail d'équipe. Il nous a permis de mettre en application les connaissances théoriques et pratiques vu au cours. Ainsi nous avons remarqué que la détection doit se faire dans un cadre bien défini au préalable suivant le type d'image (png, jpg…), moyen d'acquisition (scanné ou numérique), le type de document (administratif, politique, article...). Une fois ce cadre bien défini nous avions donc procéder à des tests afin de sélectionner les paramètres adéquates pour notre dataset.

Le véritable challenge aura été de constituer une bonne dataset correspondant aux conditions définies au départ.

Au final, nous retenons que ce projet reste améliorable et extensible par rapport à son champs d'application.