Enginyeria de Sistemes TIC

Projecte de curs de l'assignatura INFORMÀTICA

Visió artificial

Una aplicació per al reconeixement de matrícules

Alexis López
Marta Tarrés-Puertas
Sebastià Vila-Marta

Departament de Disseny i
Programació de Sistemes Electrònics

Versió 2

Índex

1 Introducció

Aquest és el guió del projecte de curs de l'assignatura d'Informàtica de l'Enginyeria de Sistemes TIC de l'EPSEM. El projecte de curs és un treball en equip que té com a finalitat el disseny (parcial) i la implementació d'una aplicació informàtica realista en la que s'apliquen els coneixements i habilitats que s'han d'haver adquirit a l'assignatura.

En aquest apartat hi trobareu una descripció de l'objectiu d'aquest projecte i les condicions, organització i terminis en que cal dur-lo a terme.

1.1 Objectiu

L'objectiu d'aquest projecte és obtenir una aplicació que permeti el reconeixement parcial de matrícules de cotxes. Parcial per que serem modestos en els objectius a fi i efecte que tingui el nivell escaient per a un projecte de curs.

Un sistema de reconeixement de matrícules és una aplicació que llegeix una imatge corresponent a una matrícula i n'extreu el seu contingut en forma de cadena de caràcters. Per exemple, si la imatge és la de la figura 1, el resultat hauria de ser la cadena `"5134FFJ"`.

Figura 1: Exemple de matrícula per processar

Una aplicació real de reconeixement de matrícula pot ser complicada. La imatge inicial pot ser la d'un cotxe sencer en mig d'una carretera. Per reconèixer la matrícula cal fer una sèrie de tractaments previs: retallar la matrícula, corregir les deformacions de la perspectiva, etc. Això és el que caldria, per exemple, si la imatge d'entrada fos similar a la de la figura 2.

Aquest projecte, però, introduirà una sèrie d'hipòtesis que permeten simplificar el problema sense perdre'n l'essència. Les simplificacions afecten a la imatge d'entrada i són les següents:

1. La matrícula usa la tipografia normalitzada.
2. La matrícula es veu frontalment, sense deformació perspectiva.
3. Únicament es veu la part de les xifres de la matrícula.

Així, el projecte només extraurà la part dels dígits per tal de simplificar i assumirà que la imatge de la matrícula és "prou bona" com per poder-la processar amb una certa tranquil·litat. La figura 3 mostra com ha de ser la imatge d'entrada del programa. És responsabilitat de l'usuari retallar les imatges fins obtenir matrícules com aquesta usant un editor d'imatges.

Figura 2: Imatge d'un cotxe i la seva matrícula. Noteu la deformació perspectiva amb que es veu la matrícula.

Figura 3: Matrícula retallada amb un editor d'imatges a punt per ser reconeguda.

2 Execució del projecte

2.1 Eines necessàries

Per dur a terme el projecte són necessàries les eines habituals que s'ha usat durant el curs: l'editor emacs, l'interpret de Python i les seves llibreries i finalment l'eina per passar els tests nosetests. A més, us caldrà tenir instal·lada la llibreria PIL i també l'aplicació gimp i l'aplicació imagemagick.

gimp us permetrà editar imatges per tal que pugueu preparar jocs de prova a partir de fotografies digitals. La llibreria PIL, [3], ofereix primitives per al processat d'imatges. En aquest projecte únicament usarem les funcions per escriure/llegir de disc i també per visualitzar les imatges. Per tal de poder visualitzar aquestes imatges ens caldrà tenir instal·lada l'aplicació imagemagick.

2.2 Metodologia de treball

La metodologia que useu per fer aquest treball és una de les claus de l'èxit. Tingueu en compte els següents aspectes:

- Apliqueu les tècniques de TDD que ja coneixeu. Totes les funcions han d'anar acompanyades de la corresponent documentació i jocs de proves que permetin comprovar exhaustivament el

programa. En aquesta aplicació, escriure els tests és un repte especialment complicat atès que el principal objecte és una imatge.

- Treballar en equip no significa que tothom ha d'estar fent la mateixa feina simultàniament sinó just el contrari: cal separar la feina de forma que cada membre de l'equip pugui treballar pel seu compte i així avançar més de pressa.
- La feina feta s'ha d'anar comprovant a mida que es fa. No podeu deixar les proves per al final del treball.
- És molt interessant que les persones de l'equip que NO han implementat un mòdul esmercin una mica de temps en llegir-lo, afegir-hi alguns casos de prova més i comprovar que funciona correctament. Això augmenta les possibilitats de detectar problemes en el mòdul.

2.3 Aspectes organitzatius

El projecte està dimensionat i dissenyat per a ésser treballat en equip. Cal que seguiu el següent esquema al treballar-hi:

1. Llegiu amb atenció individualment tot l'enunciat de punta a punta. Estudieu els coneixements previs.
2. Reuniu-vos i poseu en comú el que heu estudiat. Resoleu amb l'equip aquells punts que no heu entès.
3. Estudieu les tasques i determineu quines tasques es poden resoldre al mateix temps.
4. Planifiqueu damunt d'un calendari les feines que heu de fer per dur a terme el projecte. Anoteu quan cal tenir feta cada tasca i qui és el responsable de fer-la.
5. Repartiu-vos la tasca de documentar el projecte.

2.4 Què cal lliurar?

Per lliurar el projecte cal tenir:

1. Memòria tècnica: codi font del projecte documentat convenientment.
2. Memòria d'execució: Cal que entregueu una taula que reflecteixi l'execució del projecte. Aquesta taula ha de tenir quatre columnes: una per la data i una per a cada persona que forma l'equip. En aquesta taula hi consignareu una casella per cada dia que treballeu en el projecte. En aquesta casella indicareu molt resumidament què heu fet i quant temps hi heu dedicat. Finalment sumareu tots els temps dedicats. La taula ha de semblar aquesta:

Data	Pep	Maria	Aina
2/12		Llegir enunciat (2h)	Llegir enunciat (2h) Pensar exemple (1h)
3/12	Reunió treball (1h)	Reunió treball (1h)	Reunió treball (1h)
	1h feina	3h feina	4h feina

3 Imatges i color

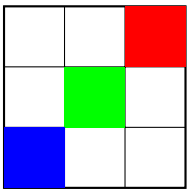
Una imatge digital, pensem en una fotografia per exemple, és des del punt de vista informàtic una matriu de *píxels*. Cada píxel representa un punt concret de la imatge i, com a tal, es representa generalment pel color d'aquest punt. Per representar un color cal anar a parar a la teoria del color.

La teoria del color és una qüestió complexa i notable. Aquí no entrarem en detalls sobre les seves peculiaritats. Si hi esteu especialment interessats o voleu entendre millor aquest aspecte no dubteu a consultar les referències de la Viquipèdia, [6], o la més orientada a aspectes artístics de Janet L. Ford, [1]. Un àmbit interessant de la teoria del color és la representació del color. A tal efecte és necessàri un model matemàtic. Hi ha diversos models matemàtics usats habitualment per representar el color que es coneixen amb el nom d'*espais de colors*, [5]. En el nostre cas usarem l'anomenat espai *RGB* o additiu, [7].

En el sistema RGB un color es representa com un punt a l'espai tridimensional $[0,1]^3 \subset \mathbb{R}^3$ en el que la primera dimensió correspon a la component R(ed), la segona a G(reen) i la tercera B(lue). Així mateix, el valor 0 per a una component indica l'absència d'aquesta component mentre que un valor 1 indica la màxima luminància en aquesta component. Així, el color blau correspon al vector $(0,0,1)$ i el color vermell al $(1,0,0)$. Podem composar multitud de colors sumant diverses proporcions de cada component. Seguint això, el color magenta correspon al vector $(1,0,1)$, el vector $(0,0,0)$ és el negre i el $(1,1,1)$ el blanc. $(0.2,0.2,0.2)$ és un gris fosc, mentre que $(0.8,0.8,0.8)$ és un gris clar i el $(0.5,0,0)$ és un vermell fosc. Moltes aplicacions informàtiques, entre elles gimp us permeten treballar amb colors definits a l'espai RGB.

Molt sovint un color RGB no es representa sobre l'espai anterior sinó sobre una aproximació en els naturals $[0,255]^3 \subset \mathbb{N}^3$. Aquesta aproximació permet representar cada component sobre un byte de memòria i és convenient pel procés informàtic.

Tornant a les imatges, doncs, convindrem que una imatge RGB és una matriu de píxels. És a dir una matriu les cel·les de la qual són tuples RGB en els naturals. Així, per exemple, a la figura 4 es pot veure una matriu de píxels a la dreta i, a l'esquerra, la imatge que li correspon.

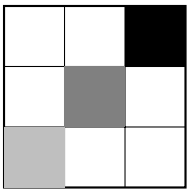


```
[
  [(255,255,255), (255,255,255), (255,0,0)],
  [(255,255,255), (0,255,0), (255,255,255)],
  [(0,0,255), (255,255,255), (255,255,255)]
]
```

Figura 4: Imatge RGB de 3×3 píxels (esquerra) i la corresponent matriu de píxels (dreta).

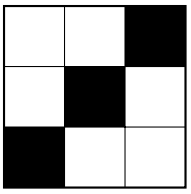
Les imatges també poden ser en escala de grisos. En aquest cas, cada píxel correspon a un valor de gris. Si usem bytes com a representació pels píxels el 0 correspon al color negre, el 255 al blanc i el 128 a un gris que té aproximadament la mateixa quantitat de blanc que de negre. La figura 5 mostra una imatge amb escala de grisos i la corresponent representació matricial.

Finalment, una imatge pot estar formada només pels colors blanc i negre. Aleshores els píxels es poden representar només per un bit. Això no obstant, per qüestions relacionades amb la facilitat d'accés a la memòria del computador, és habitual representar els píxels, com en el cas de l'escala de grisos, amb un enter que només pot prendre valors 0 o 255 indicant respectivament negre i blanc. La figura 6 correspon a una imatge en blanc i negre.



```
[
  [255, 255, 0 ],
  [255, 128, 255],
  [191, 255, 255]
]
```

Figura 5: Imatge en escala de grisos de 3×3 píxels (esquerra) i la corresponent matriu de píxels (dreta).



```
[
  [255, 255, 0 ],
  [255, 0, 255],
  [ 0, 255, 255]
]
```

Figura 6: Imatge en blanc i negre de 3×3 píxels (esquerra) i la corresponent matriu de píxels (dreta).

Tasca 1

Prepareu l'entorn de treball. A tal efecte:

- Instal·leu, en cas que no ho tingueu instal·lat, l'aplicació per visualitzar imatges imagemagick fent:

  ```
  $ sudo apt-get install imagemagick
  ```

- Instal·leu la llibreria de tractament d'imatges PIL. Com és una llibreria disponible en forma de paquet només cal que executeu l'instal·lador del sistema fent:

  ```
  $ sudo apt-get install python-imaging
  ```

- Creeu l'estructura de directoris que us sembli convenient per desenvolupar el projecte.

- Descarregueu-vos de `http://ocw.itic.cat` el tarfile que conté els patrons i el mòdul imgio que us donem implementat. Deseu-lo al directori que heu preparat prèviament.

- Desempaqueteu el tarfile executant l'ordre:

  ```
  $ tar zxvf patrons.tar.gz
  ```

  Us apareixeran una sèrie d'imatges petites, algunes matrícules d'exemple i el mòdul imgio.

- Instal·leu, en cas que no ho tingueu instal·lat, l'aplicació d'edició d'imatges gimp fent:

  ```
  $ sudo apt-get install gimp
  ```

4 Procés de reconeixement d'una matrícula

L'objectiu d'aquest apartat és donar una primera visió de com és el procediment que s'usarà en aquest projecte per a reconèixer una matrícula.

Tal com s'ha dit anteriorment, la dada d'entrada del procés és una matrícula com la de la figura 3. A partir d'aquí, l'aplicació segueix la següent estratègia:

La matrícula arriba en format RGB, el primer pas consisteix a convertir-la en una imatge en blanc i negre. Això facilitarà els processos posteriors i reduirà la seva mida a memòria. Posteriorment s'ajusta l'alçada de la imatge retallant les files superiors i inferiors de manera que encabeixi exclusivament els dígits. A continuació cal escalar verticalment la matrícula disminuint el nombre de píxels d'alçada fins que tingui la mateixa alçada que els patrons que s'usaran més endavant.

Amb la imatge de la matrícula ja preparada procedirem a separar els dígits un a un. El resultat d'aquest procés serà un conjunt de petites imatges corresponents a cadascun dels dígits de la matrícula. El darrer pas consistirà en determinat quin és el dígit de cada imatge per comparació amb uns patrons coneguts dels deu dígits possibles.

La figura 7 mostra els resultats dels diversos passos del procediment. A les imatges se'ls hi ha afegit un marc fictici per poder apreciar millor els canvis. Anant d'esquerra a dreta, la primera imatge correspon a la matrícula tal i com entra; la següent és la matrícula una vegada reescrita en blanc i negre; a continuació es veu el resultat de retallar verticalment la imatge; la segueix la matrícula escalada a l'alçada apropiada dels patrons i, finalment, es poden observar les sub-imatges corresponents als dígits de la matrícula.

Figura 7: Resultats parcials dels diferents passos en el procés de la matrícula

Tasca 2

Feu-vos amb una col·lecció fotografies de matrícules per poder provar el vostre programa. Fotografieu amb una màquina digital diverses matrícules. Procureu que la fotografia sigui nítida i la matrícula sigui centrada i sense deformar, com la de la figura 1.
Usant l'aplicació gimp que teniu en el vostre computador retalleu de les matrícules la part numèrica fins obtenir unes imatges com la de la figura 3. Aquestes figures seran el vostre joc de proves que us permetran provar les diferents fases del projecte.
Cada equip cal que tingui les seves imatges particulars.

5 Estructura de mòduls

L'estructura de mòduls del projecte és la que s'explica en el diagrama de la figura 8. En aquest diagrama una fletxa de $A \rightarrow B$ significa que el mòdul A usa el mòdul B.

Les responsabilitats principals de cada mòdul són les següents:

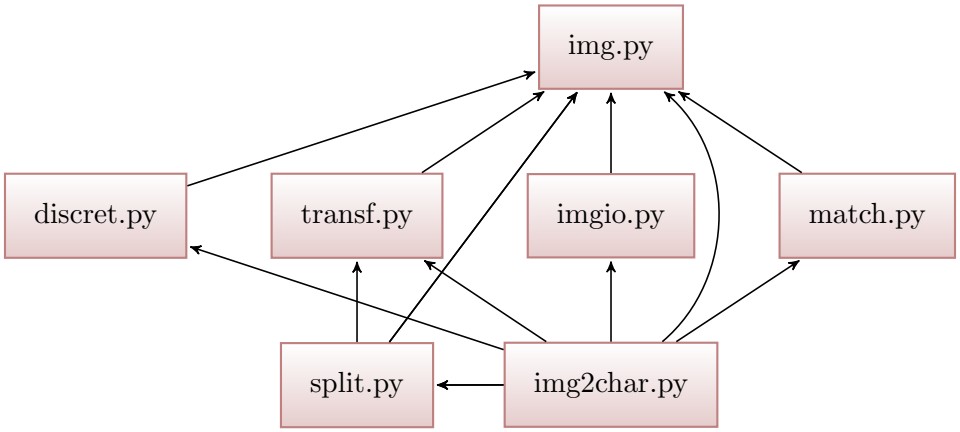


Figura 8: Diagrama de mòduls del projecte

img El mòdul conté les operacions primàries pel treball amb imatges, ja siguin imatges RGB, d'escala de grisos o de blanc i negre.

imgio Aquest mòdul és el responsable de les operacions d'entrada/sortida d'imatges: llegir imatges de disc, escriure imatges a disc i també visualitzar imatges a la pantalla. Per implementar-lo caldrà usar la llibreria PIL.

transf És el mòdul que concentra les eines per transformar imatges. En particular per escalar, és a dir per a obtenir imatges de resolucions diferents a partir d'una resolució donada i per retallar, és a dir per reduir una imatge fins que només conté els dígits que la formen.

discret És el mòdul que concentra l'algoritme de discretització, que permet calcular una imatge en blanc i negre a partir d'una imatge RGB.

split El mòdul té com a responsabilitat trencar una imatge blanc i negre en un conjunt d'imatges més petites cadascuna de les quals conté únicament un element (una xifra).

match És el mòdul que s'encarrega de determinar a quina xifra representa una imatge concreta.

img2char És el programa principal. Dels arguments que se li passen n'obté una cadena de caràcters corresponent als dígits de la matrícula.

6 El mòdul img

El mòdul img encapsula el concepte d'imatge i les seves operacions bàsiques. Com que l'aplicació treballa amb 3 tipus d'imatges —RGB, escala de grisos i blanc i negre— el mòdul ha de gestionar i saber distingir entre unes i altres. A tal efecte representarem les imatges com un tuple (T,M) en el que T representarà el tipus d'imatge i prendrà valors `'1'` per denotar imatge en blanc i negre, `'L'` si es tracta d'una imatge amb escala de grisos i `'RGB'` si es tracta d'una imatge RGB. M serà la matriu de píxels. Segons el tipus d'imatge les cel·les de la matriu seran enters o tripletes RGB. Segons aquesta representació, la imatge de la figura 4 es representa així:

```
('RGB', [[(255,255,255), (255,255,255), (255,0,0)],
         [(255,255,255), (0,255,0), (255,255,255)],
         [(0,0,255), (255,255,255), (255,255,255)]])
```

La imatge de la figura 5 es representa així:

('L', [[255,255,0], [255,128,255], [191,255,255]])

La imatge de la figura 6 es representa així:

('1', [[255,255,0], [255,0,255], [0,255,255]])

Les funcions que exporta aquest mòdul són les següents:

- null()

  Retorna una imatge nul·la. Una imatge nul·la té per representació el tuple (`'NULL'`, **None**).

- is_null(i)

  Retorna **True** sii i és una imatge nul·la.

- white_rgb(w,h)

  Retorna una imatge en format RGB amb tots els píxels blancs de $w \times h$. w és el nombre de píxels d'ample i h el nombre de píxels d'alçada.

- white_grey(w,h)

  Retorna una imatge en format escala de grisos amb tots els píxels blancs de $w \times h$. w és el nombre de píxels d'ample i h el nombre de píxels d'alçada.

- white_bn(w,h)

  Retorna una imatge en format blanc i negre amb tots els píxels blancs de $w \times h$. w és el nombre de píxels d'ample i h el nombre de píxels d'alçada.

- format(img)

  Donada una imatge img retorna el format: `'RGB'`, `'L'` o `'1'`.

- matrix(img)

  Donada una imatge img retorna la matriu de píxels corresponent.

- img(matrix, model=`'DISCOVER'`)

  S'usa per crear una imatge a partir d'una matriu de píxels matrix. Si el paràmetre model val `'DISCOVER'` la funció mira de descobrir automàticament de quin tipus d'imatge es tracta. Ho fa explorant la matriu i determinant com són els píxels. Si el paràmetre model val `'RGB'`, `'L'` o `'1'` usa aquest model després de comprovar que els píxels de la matriu són coherents amb el model.

- get_w(img)

  Retorna la dimensió w, amplada, de la imatge img.

- get_h(img)

  Retorna la dimensió h, alçada, de la imatge img.

- subimg(img,ow,oh,w,h)

  Retorna una sub-imatge de la imatge img que té l'origen a les coordenades (ow,oh) i té mides w i h. Naturalment la sub-imatge ha d'estar totalment continguda a img.

Per tal de facilitar la feina us proporcionem l'esquelet de les funcions a continuació

```
def null():
    """
    Returns the null image (Type,Matrix)==('NULL',None)
    >>> null()
    ('NULL', None)
    """

def is_null(img):
    """
    Checks if an image (Type,Matrix) is null
    >>> is_null(('NULL',None))
    True
    """

def white_rgb(w,h):
    """
    Returns an image in RGB format with all white pixels w x h
    >>> white_rgb(3,3)
    ('RGB', [[(255, 255, 255), (255, 255, 255), (255, 255, 255)], [(255, 255, 255), (255, 255, 255),
    (255, 255, 255)], [(255, 255, 255), (255, 255, 255), (255, 255, 255)]])
    """

def white_grey(w,h):
    """
    Returns an image in grey format with all white pixels w x h
    >>> white_grey(3,3)
    ('L', [[255, 255, 255], [255, 255, 255], [255, 255, 255]])
    >>> white_grey(5,2)
    ('L', [[255, 255, 255, 255, 255], [255, 255, 255, 255, 255]])
    """

def white_bn(w,h):
    """
    Returns an image in black and white format with all white pixels w x h
    >>> white_bn(3,3)
    ('1', [[255, 255, 255], [255, 255, 255], [255, 255, 255]])
    >>> white_bn(2,3)
    ('1', [[255, 255], [255, 255], [255, 255]])
    """

def format(img):
    """
    Retuns the image format
    >>> format(('1', [[255, 255], [255, 255], [255, 255]]))
    '1'
    >>> format(('L', [[255, 255, 255], [255, 255, 255], [255, 255, 255]]))
    'L'
    """

def matrix(img):
```

```
    """
    Return the pixel matrix of the image
    >>> matrix(('1', [[255, 255], [255, 255], [255, 255]]))
    [[255, 255], [255, 255], [255, 255]]
    >>> matrix(('L', [[255, 255, 255], [255, 255, 255], [255, 255, 255]]))
    [[255, 255, 255], [255, 255, 255], [255, 255, 255]]
    """

def img(m, model='DISCOVER'):
    """
    Returns the image representation format (T,m)
    >>> img([[255,255,0],[255,128,255],[191,255,255]],'DISCOVER')
    ('L', [[255, 255, 0], [255, 128, 255], [191, 255, 255]])
    >>> img([[255,255,0],[255,0,255],[0,255,255]],'DISCOVER')
    ('1', [[255, 255, 0], [255, 0, 255], [0, 255, 255]])
    """

def get_w(img):
    """
    Returns the image width
    >>> get_w(('1', [[255, 255], [255, 255], [255, 255]]))
    2
    >>> get_w( ('L', [[255, 255, 255, 255, 255], [255, 255, 255, 255, 255]]))
    5
    """

def get_h(img):
    """
    Returns the image height
    >>> get_h(('1', [[255, 255], [255, 255], [255, 255]]))
    3
    >>> get_h(('L', [[255, 255, 255], [255, 255, 255], [255, 255, 255]]))
    3
    """

def subimg(img, ow, oh, w, h):
    """
    subimg(image,column,row,widthColumns,heightRows)
    Returns a subimage from img with origin coordinates ow,oh and size w x h
    >>> subimg(('L', [[0, 0, 255], [255, 255, 255], [255, 255, 255]]),0,0,2,1)
    ('L', [[0, 0]])
    >>> subimg(('1', [[255, 255], [255, 255], [255, 255]]),0,1,2,1)
    ('1', [[255, 255]])
    >>> subimg(('1', [[255, 255], [255, 255], [0, 255]]),0,1,2,2)
    ('1', [[255, 255], [0, 255]])
    """
```

Tasca 3

Implementeu el mòdul img. Assegureu-vos que les funcions tenen els corresponents doctests i són correctes.

7 El mòdul imgio

El mòdul és el responsable de les funcions d'entrada/sortida relacionades amb les imatges. Com els formats d'emmagatzemar imatges habituals són força sofisticats usarem la llibreria PIL,[3], per implementar aquest mòdul.

Les funcions que exporta el mòdul són les següents:

- read_rgb(nomf)

  Retorna una imatge RGB llegida del fitxer de nom nomf. El fitxer pot tenir qualsevol dels formats habituals per a imatges: jpeg, png, etc.

- read_bn(nomf)

  Retorna una imatge blanc i negre llegida del fitxer de nom nomf. El fitxer pot tenir qualsevol dels formats habituals per a imatges: jpeg, png, etc.

- show(img)

  Obre una petita finestra a la pantalla en la que es mostra la imatge img. Aquesta funció és especialment important per ajudar a depurar el programa ja que permet veure les imatges que es van construint.

- save(img,nomf)

  Crea un fitxer amb nom nomf a partir de la imatge img. El fitxer podrà ser amb qualsevol dels formats habituals per a imatges: jpeg, png, etc. Recordeu que l'extensió ha de formar part del nomf.

Per tal de facilitar la feina aquestes funcions us les proporcionem a continuació:

```
import img, Image

def read_rgb(nomf):
    """
    Donat un nom de fitxer corresponent a una imatge RGB la llegeix
    i torna la imatge corresponent
    """
    image = Image.open(nomf)
    pix = image.load()
    X = image.size[0]
    Y = image.size[1]
    data = [[pix[x,y] for x in range(X)] for y in range(Y)]
    return img.img(data, 'RGB')
```

```
def read_bn(nomf):
    """
    Donat un nom d'arxiu corresponent a una imatge en blanc i negre,
    retorna la matriu d'imatge. Cada pixel serà 0 o 255.
    """
    image = Image.open(nomf).convert('1')
    pix = image.load()
    X = image.size[0]
    Y = image.size[1]
    data = [[pix[x,y] for x in range(X)] for y in range(Y)]
    return img.img(data, '1')


def show(i):
    """
    Donada una imatge, la mostra en un visualitzador a la terminal.
    Principalment serveix per a depurar el projecte.
    """
    image = Image.new(img.format(i),(img.get_w(i),img.get_h(i)))
    image.putdata([pixel for F in img.matrix(i) for pixel in F])
    image.show()


def save(i,nomf):
    """
    Donada una imatge i un nom de fitxer, crea el fitxer imatge a
    partir de la matriu.
    """
    image = Image.new(img.format(i),(img.get_w(i),img.get_h(i)))
    image.putdata([pixel for F in img.matrix(i) for pixel in F])
    image.save(nomf)
```

Tasca 4

Creeu el mòdul imgio amb les funcions proporcionades. Comproveu que funciona correctament fent un petit programa que l'usa per llegir una imatge i mostrar-la a la pantalla. Recordeu que us cal tenir instal·lada la llibreria PIL.

8 El mòdul discret

L'objectiu d'aquest mòdul és oferir una única funció que permet convertir imatges en format RGB a imatges en blanc i negre. La funció té una especificació molt simple:

- rgb_to_bn(img)

  Retorna una imatge de la mateixa dimensió de img que conté la mateixa escena però convertida a blanc i negre.

Per fer aquesta conversió la idea a aplicar és la següent:

1. A partir de la imatge en color cal calcular una imatge en escala de grisos. A tal efecte es calcula, per a cada píxel RGB, la quantitat de llum que emet, la luminància. Aquesta quantitat es traslladada a un gris proporcionalment. Aquesta imatge en escala de grisos és la luminància de la imatge original.
2. A continuació cal decidir un llindar de luminància. Per sobre d'aquest considerem que el color és blanc, per sota d'aquest és negre.
3. Una vegada se sap el llindar, cal calcular la imatge de blanc i negre a partir de la luminàncies simplement determinant quins píxels han de ser blancs i quins negres.

Pel primer pas, simplement cal tenir en compte que la luminància d'un píxel RGB que té valors (r, g, b) és el valor $L = \frac{1}{3} \cdot (r + g + b)$. Naturalment $L \in [0, 255]$.

Per calcular el llindar cal usar l'algoritme de Nobuyuki Otsu, [4]. Aquest algoritme determina un llindar que intenta capturar la màxima informació de la imatge original. Per entendre l'algoritme estudieu-vos la pàgina del Dr. Greensted que trobareu a [2].

A continuació segueix el detall de les funcions a implementar.

```
def rgb_to_lum(pixel):
    """
    Returns the luminance level off a RGB pixel
    >>> rgb_to_lum((255,255,255))
    255
    >>> rgb_to_lum((255,0,0))
    85
    """


def luminance_img(i):
    """
    Transforms a RGB image to a L image using luminance
    >>> luminance_img(('RGB', [[(255, 255, 255), (255, 255, 255), (255, 255, 255)], [(255, 255, 255),
            (255, 255, 255), (255, 255, 255)], [(255, 255, 255), (255, 255, 255), (255, 255, 255)]]))
    ('L', [[255, 255, 255], [255, 255, 255], [255, 255, 255]])
    """


def histogram(i):
    """
    Histogram of grey values of L image i
    >>> histogram(('L', [[255, 255, 255, 255], [255, 255, 255, 255]]))
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8]
    """

def get_threshold(h):
    """
    Get the (index of) two max elements of the histogram
    Usem Otsu thresholding algorithm.
    http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html
    """

def rgb_to_bn(i):
    """
    Transforms a RGB image to a 1 image using luminance, histogram and get_threshold
    >>> rgb_to_bn(('RGB', [[(255, 255, 255), (255, 255, 255), (255, 255, 255)],[(255, 255, 255),
                    (255, 255, 255), (255, 255, 255)], [(255, 255, 255), (255, 255, 255), (255, 255, 255)]]))
    ('1', [[255, 255, 255], [255, 255, 255], [255, 255, 255]])
    """
```

Tasca 5

Estudieu amb cura les referències sobre l'algoritme d'Otsu fins que l'entengueu bé. Implementeu el mòdul usant aquest algoritme. Proveu-lo sobre imatges de diferent natura per assegurar-vos que funciona com cal.

9 El mòdul transf

Aquest mòdul encapsula diverses funcions que permeten fer transformacions a les imatges. Les funcions són les següents:

- vtrim(img)

  Donada una imatge img en blanc i negre, retorna l'imatge resultant de retallar-la verticalment. Per retall vertical s'entén extreure la sub-imatge que, en la dimensió vertical encabeix de forma exacta els dígits. Amb això s'aconsegueix que els dígits de la matricula quedin ben "enquadrats" verticalment. Si la imatge només conté blancs, retorna una imatge nul·la.

- htrim(img)

  Fa una feina similar a la funció vtrim() però en la direcció horitzontal.

- scale(img,h)

  Escala homogèniament img fins que la seva alçada és h. S'usarà principalment per reduir la mida d'una imatge.

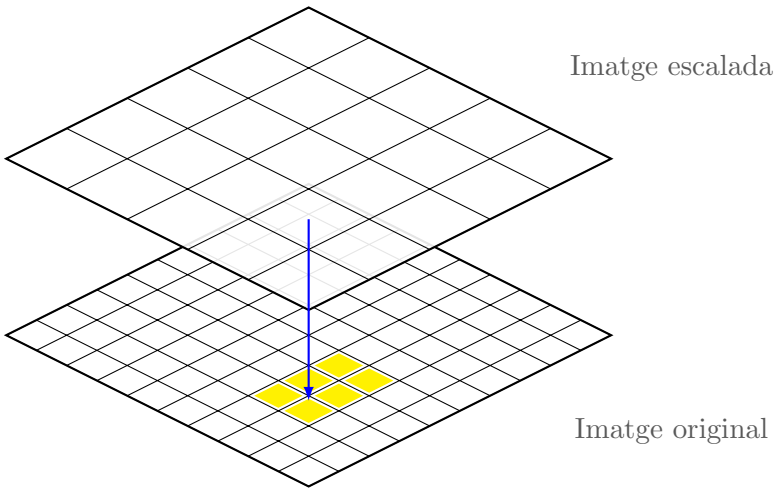


Figura 9: Esquema per a la reducció de resolució.

Per dur a terme l'escalat optem per la forma més simple possible. Únicament cal fer un petit càlcul geomètric. Si la imatge original té dimensions $W \times H$ i l'alçada que ha de tenir un com escalada és d'h, podem definir el següent factor d'escalat vertical:

$$f_h = \frac{H}{h}$$

Com la imatge ha de conservar les proporcions, el factor que s'aplicarà a l'escalat horitzontal ha de ser el mateix.

Aleshores el píxel de coordenades (a, b) de la imatge escalada resolució es projecta sobre el píxel $(\lfloor af_h \rfloor, \lfloor bf_h \rfloor)$ de la imatge original. Vegeu l'esquema a la figura 9.

Per calcular la imatge escalada, només cal recórrer els seus píxels. A cada píxel de la imatge escalada se li assigna el color del píxel corresponent de la imatge original. Per saber quin píxel de la imatge original correspon a un de la imatge escalada s'aplica el factor d'escalat tal i com s'ha dit abans.

Per tal de facilitar la feina us proporcionem l'esquelet de les funcions a implementar a continuació.

```
def scale(src, h):
    """
    Scale image src taking into account height h preserving ratio aspect
    >>> scale(('RGB', [[(255, 255, 255), (255, 255, 255), (255, 255, 255)], [(255, 255, 255),
                (255, 255, 255), (255, 255, 255)], [(255, 255, 255), (255, 255, 255), (255, 255, 255)]]),2)
    ('1', [[(255, 255, 255), (255, 255, 255)], [(255, 255, 255), (255, 255, 255)]])
    >>> scale(('RGB', [[(0, 0, 0), (255, 255, 255), (255, 0, 0)], [(255, 255, 255), (0, 255, 0),
                (255, 255, 255)], [(0, 0, 255), (255, 255, 255), (255, 255, 255)]]),2)
    ('1', [[(0, 0, 0), (255, 0, 0)], [(0, 0, 255), (255, 255, 255)]])
    """


def vtrim(i):
    """
    Returns vertical trimmed image
    >>> vtrim(('1',[[255,255,0],[255,0,255],[0,255,255]]))
```

```
    ('1', [[255, 255, 0], [255, 0, 255], [0, 255, 255]])
    >>> vtrim(('1',[[255,255,255],[255,0,255],[0,255,255]]))
    ('1', [[255, 0, 255], [0, 255, 255]])
    """

def htrim(i):
    """
    Returns horizontal trimmed image
    >>> htrim(('1',[[255,255,0],[255,0,255],[0,255,255]]))
    ('1', [[255, 255, 0], [255, 0, 255], [0, 255, 255]])
    >>> htrim(('1',[[255,255,255],[255,0,255],[0,255,255]]))
    ('1', [[255, 255], [255, 0], [0, 255]])
    """
```

Tasca 6

Implementeu el mòdul. Afegiu els doctests escaients usant imatges de mida petita i comproveu que són correctes. Aneu amb compte amb l'escalat d'imatges: cal entendre bé el procediment i ser curós per implementar-lo com cal.

10 El mòdul split

L'objectiu d'aquest mòdul és trencar una imatge d'una matrícula ja preparada (retallada verticalment, redimensionada i en blanc i negre) en una sub-imatge per cada dígit. Per exemple, si considerem la matrícula de la figura 1 la funcionalitat d'aquest mòdul ha de permetre trencar-la en dues sub-imatges. La primera que conté el dígit de més a l'esquerra, vegeu la figura 10. La segona que conté la resta de la imatge, vegeu la figura 11.

L'aplicació repetida d'aquest procediment permet separar els dígits de la matrícula inicial.

Figura 10: Primer digit separat

Figura 11: La resta de la matrícula

Aquest mòdul ofereix una funció amb la següent especificació:

- split_digit(img)

  Aquesta funció rebrà una imatge img en blanc i negre retallada verticalment i retorna una tupla (D,R) en la que D és una imatge amb el dígit de més a l'esquerra i R és la resta de la imatge. La imatge corresponent al dígit extret D es retorna convenientment retallada en la direcció horitzontal. La resta R esdevé una imatge nul·la quan s'han extret tots els dígits.

Tasca 7

Implementeu el mòdul i afegiu els doctests escaients per garantir el seu correcte funcionament.

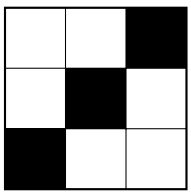
11 El mòdul match

Aquest mòdul és el responsable del reconeixement dels dígits. Aquest reconeixement es farà en base a una col·lecció de patrons donada, és a dir una col·lecció d'imatges estàndard dels deu dígits que es poden reconèixer.

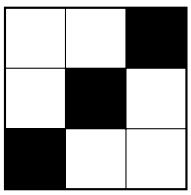
L'estratègia de treball és senzilla. Assumim que el patró del dígit i és la imatge P_i, on $i \in [0, 9]$. Per determinar quin dígit conté la imatge X, calculem la semblança entre X i P_i per a tots els patrons $i \in [0, 9]$. Algun d'aquests patrons serà el més semblant. Considerarem que I conté el dígit corresponent al patró més semblant. Si la funció de semblança entre dues imatges qualssevol de la mateixa mida A i B l'escrivim com $sim(A, B)$, llavors el dígit que conté la imatge X és exactament el $i \in [0, 9]$ tal que $\forall j : j \in [0, 9] : sim(X, Pi) \leq sim(X, Pj)$.

La dificultat rau ara en com podem calcular $sim(A, B)$. Hi ha varies tècniques per poder quantificar la semblança entre dues imatges. En aquest cas se'n farà servir una de ben senzilla. Es tracta de comptar quants píxels iguals comparteixen les imatges A i B. Com més semblants siguin, més píxels compartiran. Entenem que A i B comparteixen un píxel (x, y) si $A[x, y] = B[x, y]$.

Per exemple, la figura 12 mostra dues imatges amb les seves matrius corresponents. Si calculem el nombre de blancs que coincideixen en les dues imatges obtindrem el valor 6. La semblança entre una i altra imatge és doncs de 9. En canvi, entre les imatges de la figura 13 la similitud és de 5. Podem dir doncs que les imatges de la figura 12 són més semblants que les de la figura 13.



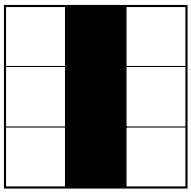
```
[
  [255, 255, 0 ],
  [255, 0, 255],
  [ 0, 255, 255]
]
```



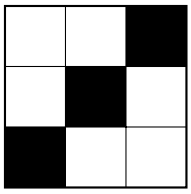
```
[
  [255, 255, 0 ],
  [255, 0, 255],
  [ 0, 255, 255]
]
```

Figura 12: Dues imatges iguals amb les seves corresponents matrius.

En comparar la imatge provinent de la matrícula amb els patrons cal tenir en compte el següent aspecte: els patrons i els dígits tindran la mateixa altura, atès que els haurem normalitzat prèviament, però no tindran necessàriament la mateixa amplada. Per tant hi haurà imatges i patrons amb amplàries diferents. Com es calcula la funció de semblança en aquest cas?



```
[
  [255, 0, 255],
  [255, 0, 255],
  [255, 0, 255]
]
```



```
[
  [255, 255, 0 ],
  [255, 0, 255],
  [ 0, 255, 255]
]
```

Figura 13: Dues imatges diferents amb les seves corresponents matrius.

Observeu la figura 14. A la figura hi ha una imatge de color gris i una altre de color blau que podeu veure a la fila superior. Per calcular la semblança entre una i altra imatge calculem les diverses semblances que s'obtenen d'anar desplaçant la imatge petita, blava, sobre la gran, grisa. A cada posició s'obté una semblança diferent. En el cas de la figura 14 i mirant d'esquerra a dreta, les semblances valen 5, 6, 5 i 6. De totes elles ens quedem amb la semblança més gran i per tant conclouríem que la semblança entre la imatge blava i la gris és de 6.

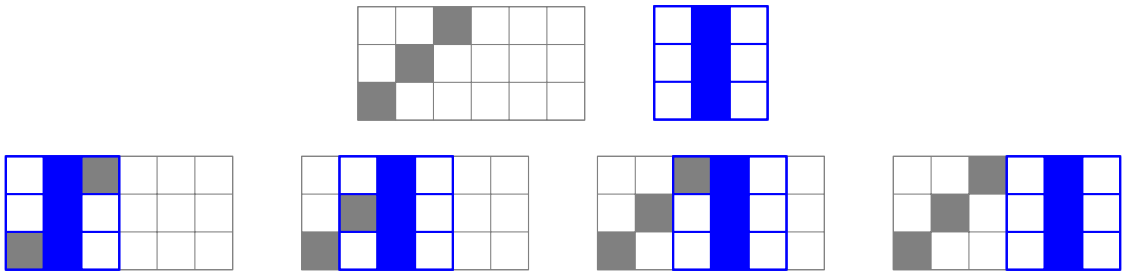


Figura 14: Passos en el càlcul de la semblança entre imatges d'ample diferent

Aquest mòdul implementa les següents funcions:

- load_patterns(prefix)

  Aquesta funció rep com a paràmetre el prefix dels noms dels fitxers que contenen els patrons dels dígits i retorna la llista d'imatges corresponent als patrons dels dígits ordenats de 0 a 9. Per exemple, si l'argument és `patro` voldrà dir que els arxius dels patrons que s'hauran de llegir s'anomenaran: `patro_0.jpeg`, `patro_1.jpeg`, ..., `patro_9.jpeg`.

- match(img, patlst)

  Donada una llista de patrons patlst i una imatge img retorna un enter que correspon amb el dígit més semblant d'acord amb els conjunt de patrons usat. La imatge img ha de tenir la mateixa alçada que els patrons.

Tasca 8

Implementeu el mòdul seguint l'estratègia que s'ha explicat. Proveu-lo per assegurar que funciona com s'espera.

12 El mòdul img2char

Aquest mòdul correspon al programa principal. A l'hora d'executar el programa principal especificarem el nom de l'arxiu corresponent a l'imatge de la matricula i el prefix del nom dels patrons. El programa haurà d'escriure a la pantalla el número de matrícula que consta a la imatge.

El que segueix és un exemple d'invocació d'aquest programa:

```
$ python img2char.py carpeta_patrons/patro matricula.jpeg
```

En aquest exemple , `matricula.jpeg` és la imatge de la matrícula que volem analitzar i `patrons/patro` el prefix dels arxius dels patrons (per tant es buscarien els patrons `patrons/patro_1.jpeg`, `patrons/patro_2.jpeg`, ...).

Recordeu que per processar la matrícula cal:

1. Obtenir la llista de patrons que seran imatges en format blanc i negre.
2. Llegir la imatge de la matrícula.
3. Convertir la matrícula a blanc i negre.
4. Ajustar l'alçada de la matrícula retallant les franges blanques que puguin existir.
5. Escalar la matricula a fi i efecte que l'alçada coincideixi amb la dels patrons.
6. Extreure els dígits de la matrícula i, simultàniament, determinar a quina xifra representen mitjançant el matching.
7. Mostrar l'enter que correspon a la matrícula.

Tasca 9

Implementeu i comproveu el funcionament del programa principal.

Tasca 10

Prepareu la documentació del projecte i el tarfile per lliurar-lo seguint el procediment que ja us hauran indicat.

Referències

[1] Janet Lynn Ford. *Color Theory*. 2012. URL: http://www.worqx.com/color (cons.).

[2] Andrew Greensted. *Otsu Thresholding*. Jun. 2010. URL: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html (cons.).

[3] Fredrik Lundh i Matthew Ellis. *PIL. Python Imaging Library Handbook*. 2005. URL: http://www.pythonware.com/library/pil/handbook/index.htm (cons.).

[4] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". A: *IEEE Transactions on Systems, Man and Cybernetics* 9.1 (1979), pàg. 62 - 66. DOI: 10.1109/TSMC.1979.4310076. URL: http://dx.doi.org/10.1109/TSMC.1979.4310076.

[5] Wikipedia. *Color Space — Wikipedia, The Free Encyclopedia*. 2012. URL: http://en.wikipedia.org/wiki/Color_space (cons.).

[6] Wikipedia. *Color Theory — Wikipedia, The Free Encyclopedia*. 2012. URL: http://en.wikipedia.org/wiki/Color_theory (cons.).

[7] Wikipedia. *RGB Color Space — Wikipedia, The Free Encyclopedia*. 2012. URL: http://en.wikipedia.org/wiki/Color_theory (cons.).