Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2014

IIC 1103 - Introducción a la Programación

Tarea 3

Objetivo General

Esta tarea tiene como objetivo que te enfrentes a un desafío de programación que te permita practicar los conocimientos adquiridos de Python, centrados principalmente en el tópico de *listas*. Para el logro de este objetivo deberás implementar en Python de el programa de procesamiento de imágenes llamado *InstaIntro*.

Introducción

La fotografía digital usa un conjunto de sensores electrónicos para capturar la imagen enfocada por el lente de una cámara. Estos sensores perciben la intensidad de luz y convierten la imagen enfocada en una matriz (lista de listas) de pixeles (Figura 1). Cada pixel es representado por una tupla de tres números enteros entre 0 y 255. El primero de estos números representa la intensidad de rojo, el segundo la intensidad verde y el tercero la intensidad de azul. Por ejemplo, la tupla (0, 150, 0) representa un pixel de color verde opaco, mientras que la tupla (0, 255, 255) representa un pixel de color cyan (color que resulta de mezclar el verde con el azul).

Dada una imagen **A** en su representación como matriz, consideraremos que el elemento **A**[0][0] corresponde al pixel de la ezquina superior izquierda. La primera coordenada corresponderá luego al número de fila (de arriba hacia abajo) y la segunda al número de columna (de izquierda a derecha), tal como se muestra en la Figura 1.

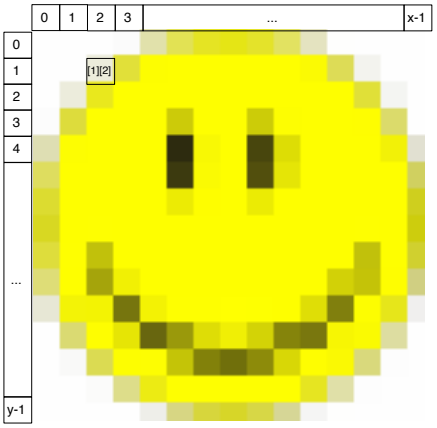


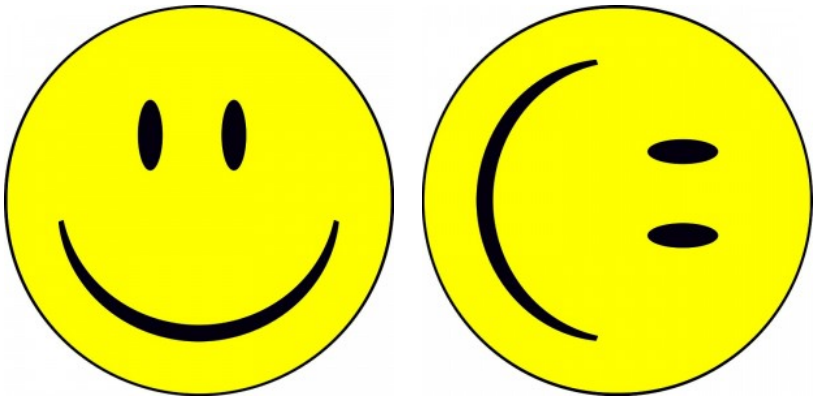
Figura 1: Imagen digital como un matriz de pixeles.

Enunciado

En esta tarea deberás escribir un programa que permita al usuario abrir una imagen en formato *gif* y aplicarle las siguientes operaciones:

Girar

Gira la imagen 90 grados en el sentido de las manecillas del reloj.

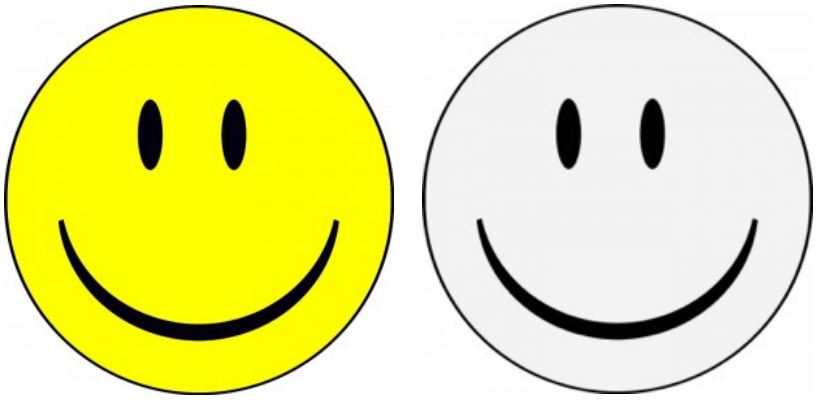


(a) Imagen original (b) Imagen girada 90 grados

Figura 2: Girar la imagen.

Escala de Grises

Convierte una imagen de color a otra en escala de grises. Para que una imagen sea vea en tonos de gris se requiere que los tres componentes básicos del color (rojo, verde, azul) tengan más o menos la misma intensidad. Por lo tanto, podemos decir que si queremos convertir un pixel a su equivalente en escala de grises bastaría con cambiar la intensidad de cada color del pixel por el promedio de los tres colores básicos. Por ejemplo, el pixel $(255, 0, 0)$ será convertido al $(85, 85, 85)$.

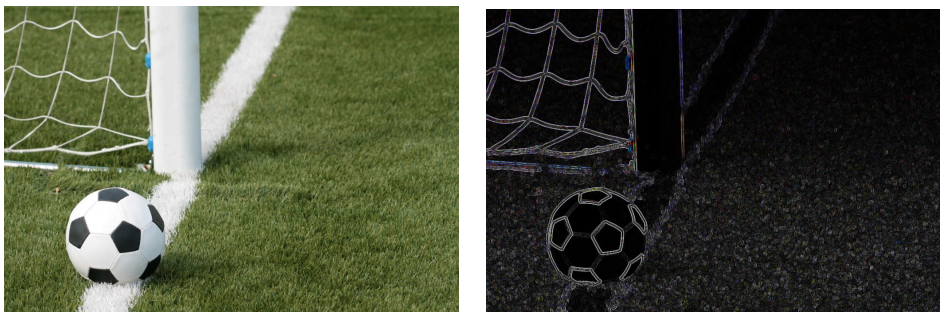


(a) Imagen original (b) Imagen en escala de grises

Figura 3: Convertir una imagen a escala de grises.

Bordes

Genera una imagen en escala de grises con líneas blancas que representan los bordes de la imagen original.



(a) Imagen original (b) Bordes de la imagen

Figura 4: Obtener los bordes de una imagen.

Para calcular esta imagen, primero se transforma la imagen original a su representación en escala de grises, que denominaremos $\mathbf{A}$. Luego se generan dos matrices: una para detectar los bordes horizontales y otra para detectar los bordes verticales. Estas dos matrices se denotan $\mathbf{G}_x$ y $\mathbf{G}_y$, y se definen como:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad \text{y} \quad \mathbf{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A},$$

donde $*$ es el operador de *convolución*. Este operador toma una matriz $\mathbf{B}$ de 3 x 3 y una matriz de pixeles $\mathbf{C}$, y genera una nueva matriz de pixeles $\mathbf{C}'$, donde el pixel $\mathbf{C}'[i][j]$ es definido como:

$$\mathbf{C}'[i][j] = \sum_{k=0}^{2} \sum_{\ell=0}^{2} \mathbf{B}[k][\ell] \cdot \mathbf{C}[i+k-1][j+\ell-1].$$

Cabe notar que cada entrada de $\mathbf{C}$ es una tupla y cada entrada de $\mathbf{B}$ es un número, por lo que la multiplicación de una entrada de $\mathbf{C}$ por una entrada de $\mathbf{B}$ es la multiplicación usual entre un vector y un escalar.

Para calcular el operador convolución sobre los pixeles en los extremos, no se toman en cuenta las coordenadas que se *salen* de la imagen. Por ejemplo, $\mathbf{C}'[0][0]$ se calcularía sin considerar la primera fila ni la primera columna de $\mathbf{B}$:

$$\mathbf{C}'[0][0] = \mathbf{B}[1][1] \cdot \mathbf{C}[0][0] + \mathbf{B}[1][2] \cdot \mathbf{C}[0][1] + \mathbf{B}[2][1] \cdot \mathbf{C}[\mathbf{1}][\mathbf{0}] + \mathbf{B}[2][2] \cdot \mathbf{C}[1][1].$$

Dado un pixel $p = (r, g, b)$, definimos p^2 como (r^2, g^2, b^2) y $\sqrt{p}$ como $(\lfloor\sqrt{r}\rfloor, \lfloor\sqrt{g}\rfloor, \lfloor\sqrt{b}\rfloor)$. A partir de las imágenes $\mathbf{G_x}$ y $\mathbf{G_y}$ se genera una matriz G cuyo elemento en la posición $[i][j]$ se calcula como

$$\mathbf{G}[i][j] = \sqrt{\mathbf{G_x}[i][j]^2 + \mathbf{G_y}[i][j]^2}.$$

De acuerdo a lo anterior, la matriz $\mathbf{G}$ contiene en cada entrada una tupla de tres números, los cuales podrían ser mayores que 255. Para transformar esta matriz a una imagen debemos 'normalizar' las tuplas que contienen números mayores a 255. Dada una tupla $t = (t_1, t_2, t_3)$ definimos el factor de normalización de dicha tupla como

$$n_t = \frac{\text{máx}\{t_1, t_2, t_3, 255\}}{255}$$

Teniendo el factor de normalización, se define el pixel asociado a la tupla t como

$$p(t) = \left(\frac{t_1}{n_t}, \frac{t_2}{n_t}, \frac{t_3}{n_t}\right)$$

Finalmente, generamos una imagen $\mathbf{H}$ a partir de la normalización de las tuplas de la matriz $\mathbf{G}$. Más precisamente, el pixel en la posición $[i][j]$ de la imagen $\mathbf{H}$ (imagen que muestra los bordes de $\mathbf{A}$) se calcula como

$$\mathbf{H}[i][j] = p(\mathbf{G}[i][j]).$$

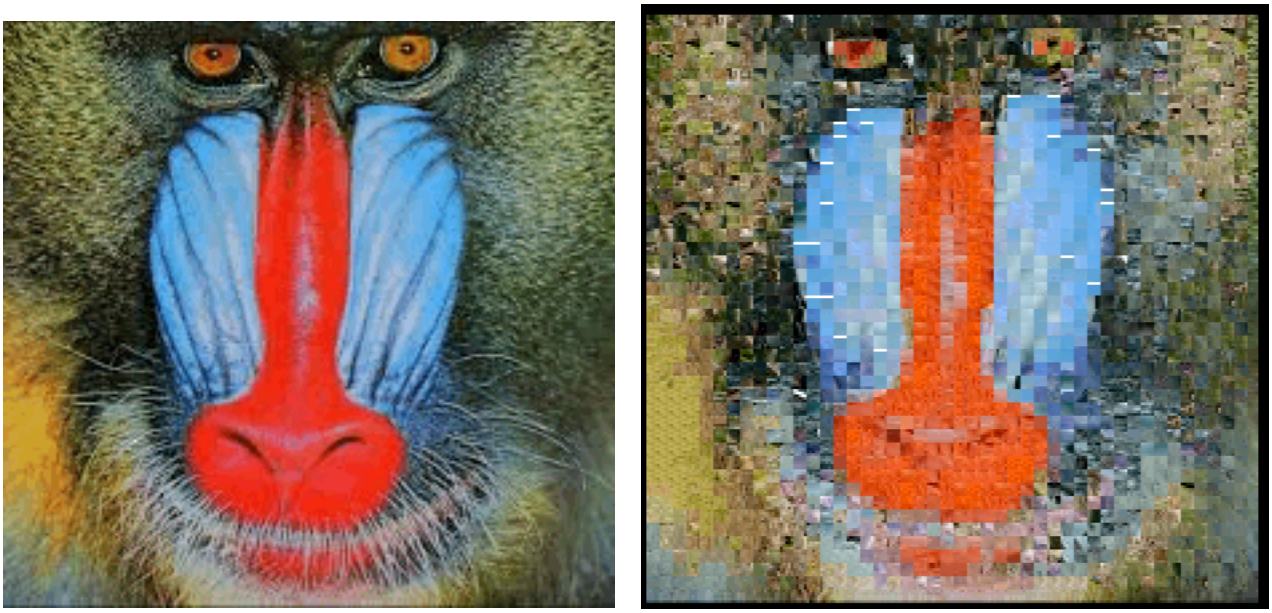
Mosaico

La carpeta de tu tarea deberá incluir una subcarpeta llamada *mosaicos* que contiene un buen número de imágenes de tamaño 5 x 5 pixeles.

El operador Mosaico consiste en generar una nueva imagen reemplazando porciones de la imagen original por la imagen de 5 x 5 que más se parece a dicha porción.

Más específicamente, dada una imagen base (de tamaño arbitrario), deberás dividirla en porciones cuadradas de 5 x 5 pixeles (puedes asumir que las dimensiones de tu imagen son múltiplos de 5). Posteriormente, para cada uno de esas porciones deberás elegir la imagen de la carpeta mosaicos que más se parece a la porción, y reemplazar esa porción por la imagen elegida.

Para calcular la similitud entre una imagen I_1 y otra imagen I_2 se considera la suma de las diferencias entre cada pixel de I_1 y el pixel que corresponde a la misma posición en I_2; a su vez, la diferencia entre el pixel $p1 = (r_1, g_1, b_1)$ y $p2 = (r_2, g_2, b_2)$ es el resultado de $(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2$.



(a) Imagen original

(b) Imagen como un mosaico

Figura 5: Transformar una imagen a un mosaico de porciones de imágenes.

Librería instaintro_gui

Para llevar a cabo tu tarea, cuentas con la librería instaintro_gui, que te permitirá manejar el programa a través de una interfaz gráfica. Al iniciar tu programa se abrirá una interfaz gráfica que tiene los elementos necesarios para que funcione el sistema de edición de imágenes (ver Figura 6).

Para que tu programa pueda interactuar con el usuario correctamente, la librería `instaintro_gui` ofrece las siguientes instrucciones:

- `obtener_pixeles()`: Retorna la matriz de pixeles de la imagen que fue cargada usando el botón `Abrir`.
- `actualizar_imagen(pixeles)`: Actualiza la imagen usando la matriz de pixeles que recibe como parámetro.
- `esperar_click()`: Al llamar a esta instrucción el programa se detiene hasta que el usuario haga click en alguno de los botones de la interfaz que no sea el botón Abrir. Luego retorna el nombre en minúsculas del botón que se presionó. Por ejemplo, si el usuario presionó el botón `Gris` entonces esta función retornará la cadena de texto `gris`.

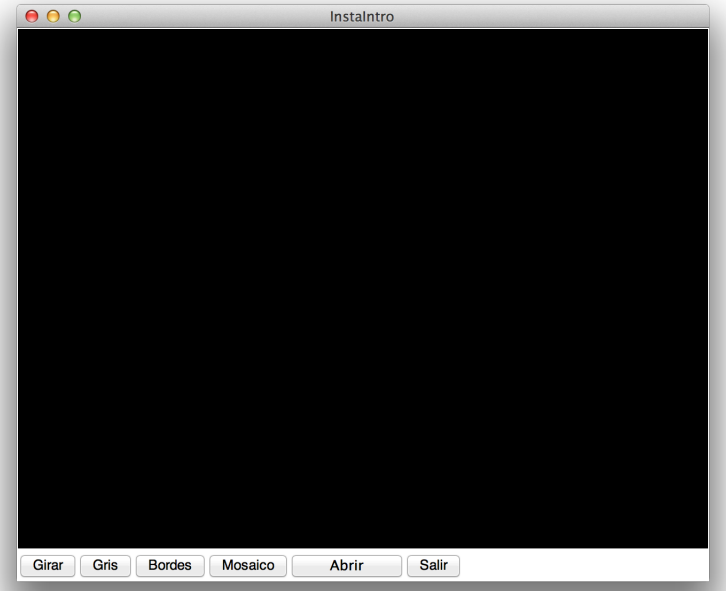


Figura 6: Ventana principal.

- `obtener_imagenes_mosaico()`: Al llamar a esta instrucción el programa procesará aquellas imágenes de tipo GIF (`.gif`) que estén contenidas en la carpeta `mosaico`. Luego de esto, la función retornará la lista de imágenes procesadas. Ten en cuenta que cada imagen es una matriz de pixeles.
- `salir()`: Ofrece al usuario salir del programa y, en caso de que el usuario acepte, cierra el programa.

Configuración

Para que tu proyecto funcione correctamente, debes tener en una misma carpeta tu proyecto junto con los archivos que se encuentran junto a este enunciado en la página del curso:

- `instaintro_gui.py`
- `mosaicos`

Además, el archivo de tu proyecto debe cumplir con la plantilla que se muestra en el siguiente fragmento de código y que también podrás encontrar en la página del curso.

```
import instaintro_gui

def tarea():
    #Aqui empieza tu tarea


app = intro_gui.Application("tarea")
app.title('InstaIntro')
app.loadProgram(tarea)
app.start()
```

Guarda tu proyecto con el nombre `tarea3_nroalumno_seccion_n.py` donde debes reemplazar `nroalumno` por tu número de alumno y `n` por el número de tu sección. Por ejemplo, Juan Pérez de la sección 1 con número de alumno 12345, guardará su proyecto con el nombre `tarea3_12345_seccion_1.py`.

Indicaciones Generales

- Recuerda que la tarea es completamente individual, y que el hecho de compartir código es sancionado con nota `1,1` final en la asignatura.
- La entrega de esta tarea es a través del SIDING en la sección cuestionario. Podrás subir tu tarea las veces que creas necesario mientras el plazo de entrega sea válido.
- La fecha última de entrega es el miércoles 22 de octubre hasta las 23:59 PM. No esperes hasta última hora, pues el buzón (cuestionario) se cierra automáticamente a esa hora. No se aceptarán entregas atrasadas ni entregadas por otros medios.
- Revisa bien lo que entregas, aunque esta vez podrás subir tu tarea ilimitadas veces, la versión final a corregir será la última que hayas subido.
- Solo podrás usar las librerías que estén instaladas por defecto en Python.
- En caso de dudas, recurre al foro del curso (`edx.ing.puc.cl`). Es probable que ya encuentres una respuesta, o en caso contrario ayudarás a resolver las dudas de tus compañeros.