Proyecto 1 Procesamiento de Imágenes

*Aguilar Antonio, Espinoza Marco, Vargas Mario, Zeledón Freddy*

*Abstract:*

*Índice de términos*—

# Nomenclatura

# Introducción

ESTE proyecto consiste en la elaboración de un mecanismo digital que permita emular el comportamiento de una cámara PTZ, de manera que el sistema sea capaz de capturar una imagen de Full-HD o HD, tomar un segmento de la misma con un tamaño y posición particular y escalarlo al tamaño final de 300x300 pixeles. Para ello se diseñó un software a través de C++ utilizando las librerías de OpenGL, SDL y GUCVIEW, los cuales permiten la manipulación de la imagen que se encuentra en una alta resolución para poder segmentarla de forma que pueda ser rotada, trasladada y escalada en una resolución menor que pueda ser interpretada por nuestro sistema.

# Cámara PTZ.

<Incluir explicacion de Camara PTZ>

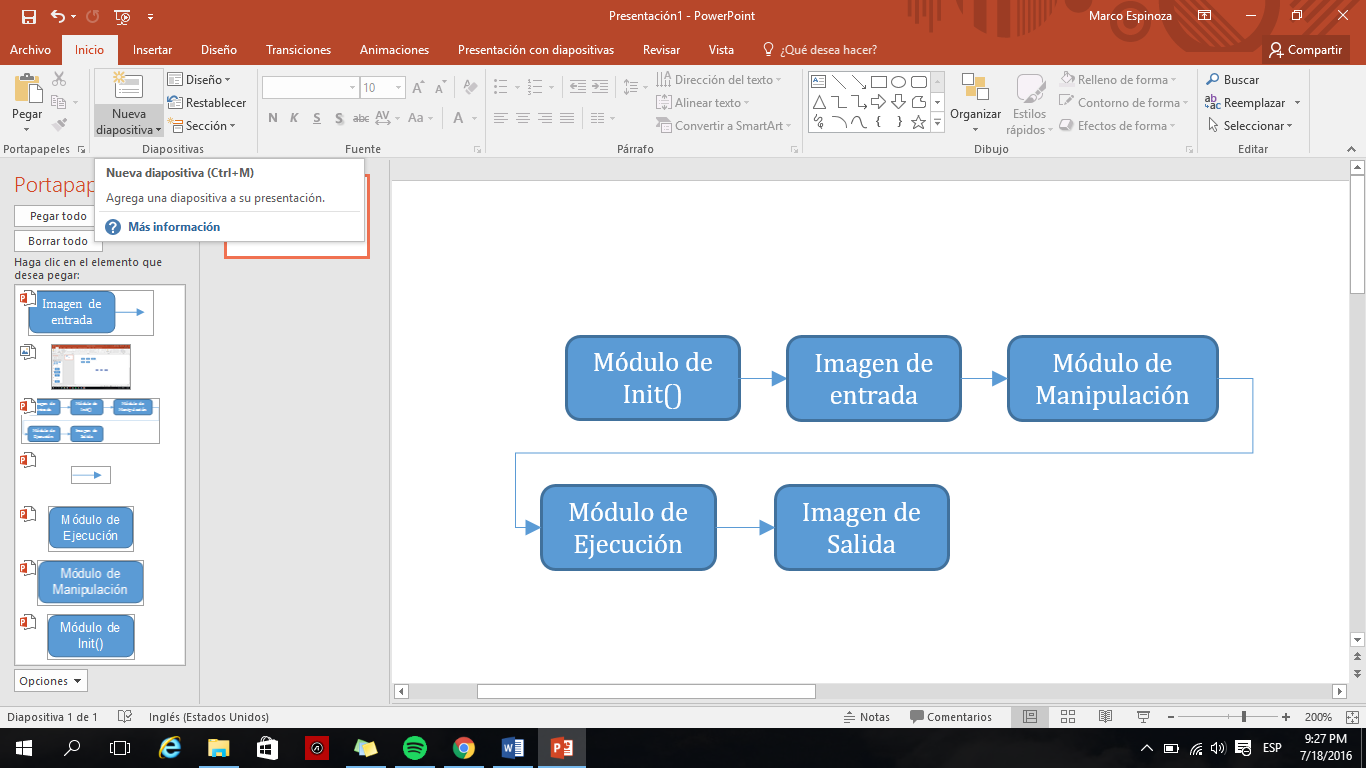
# OpenGL and SDL

<Explicacion acerca de OpenGL y SDL>

# Sistema Implementado

Para la elaboración de la solución, se utilizó las librerías de OpenGL y SDL, que facilitaron la manipulación de la imagen de entrada, tal que la imagen de salida puede cumplir con los requisitos del sistema.

Para ello se desarrolló la solución que se muestra a continuación:



**Figura 1.** Diagrama de bloques de la solución implementada.

La imagen de entrada estará en una resolución Full HD o HD, primero se debe de inicializar los métodos de las librerías de openGL y SDL requeridos para poder cargar la imagen adecuadamente.

Una vez cargada la imagen, el módulo de manipulación se encarga de recibir los argumentos del usuario para aplicarlos a la imagen original.

Como se explica en la siguiente sección, la matriz original es multiplicada por la matriz generada a través de los argumentos ingresados por el usuario, y junto con el módulo de ejecución la imagen resultante es mostrada en pantalla.

Además, los argumentos son ingresados por el usuario en tiempo real, permitiendo manipular la imagen según los requerimientos necesarios.

# Justificación de la solución

En la sección anterior, se explicó el sistema implementado que permitió la solución del proyecto. En esta sección se explicará cada uno de los módulos en detalle, para justificar la solución propuesta.

# *Módulo Init:*

En este módulo se inicializan los métodos necesarios para poder hacer la captura del video y la carga de la imagen de entrada, así como también se especifican las configuraciones de entrada.

A continuación, se explican cada uno de los métodos requeridos para poder cargar la imagen apropiadamente:

*create\_v4l2\_device\_handle(device) and v4l2core\_set\_capture\_method():*

Estos método se encargan de inicializar el dispositivo para la captura del vídeo a través de la cámara. En caso de que el dispositivo no esté correctamente configurado, la ejecución del programa terminará.

El método v4l2core\_set\_capture\_method se encarga también de inicializar el manejador del dispositivo de captura.

*glEnable(GL\_TEXTURE\_2D):*

Debido a que la entrada del sistema es una imagen 2D, este argumento permite la habilitación de las texturas en 2 dimensiones.

*glEnable(GL\_DEPTH\_TEST):*

Permite realizar comparaciones de profundidad y actualizarlos en el buffer respectivo. Este argumento es requerido para nuestro proyecto debido a que el tamaño de la imagen será modificado en tiempo real.

*glEnable(GL\_LIGHTING):*

Debido a que, para nuestra solución no se utilizó el vertex shader, este argumento permite controlar el vertex color.

*glEnable(GL\_LIGHT1):*

El mismo es utilizado para la evaluación de las ecuaciones de luminosidad.

*glMatrixMode(GL\_PROJECTION):*

Se encarga de aplicar operaciones matriciales a la pila de matrices.

*glMatrixMode(GL\_MODELVIEW):*

La misma se encarga de aplicar operaciones matriciales subsecuentes a la pila de matrices de modelview.

*glLoadIdentity();*

Se encarga de reemplazar la matriz actual con la matriz de identidad.

*LoadTexture():*

Este método se encarga de cargar la imagen de entrada y aplicar la textura a la misma.

*v4l2core\_prepare\_new\_format() and v4l2core\_prepare\_new\_resolution():*

El primer método se encarga de convertir el formato de la imagen capturada a través de la cámara a JPEG, de esta manera se pueden aplicar las métodos de las librerías de openGL a dicha imagen.

El segundo método se encarga de preparar la imagen para la resolución de salida requerido por el sistema, que en nuestro caso es de 300x300 pixeles.

*v4l2core\_start\_stream()*

Este método permite detener el streaming del vídeo para poder utilizar la imagen para ser manipulada a través del módulo de manipulación,

*Módulo de Manipulación:*

En el mismo se utiliza como entrada teclas respectivas del teclado para aplicar a la imagen y así tener como salida una imagen en una resolución de 300x300. Seguidamente, se explica cada una de las teclas de entrada y el resultado en la imagen de salida:

*Izquierda, Derecha, Arriba y Abajo*:

Permite moverse hacia la izquierda, derecha, arriba o abajo respectivamente de la imagen original, para tener como salida la imagen desplazada hacia la dirección deseada.

Para ello se utilizó el método *glTranslatef*, la cual tiene tres argumentos como entrada (x, y, z), de manera que la matriz original es multiplicada por esta nueva matriz, para tener como matriz resultante la nueva ubicación de la imagen.

*Tecla q y tecla e:*

Permite la rotación de la imagen, para ello se utiliza el método *glRotate*, el cual tiene 4 argumentos de entrada, en vez de los 3 que tenía el método de *gltranslatef*. Debido a que este método tiene el ángulo de rotación como argumento (ángulo de rotación, x, y, z) que produce una rotación de un ángulo alrededor del vector, de manera que la matriz actual es multiplicada por una matriz de rotación, de manera que la matriz resultante es la imagen con la rotación aplicada.

*Tecla i y tecla o:*

Utilizados para manipular el tamaño de la imagen, el cual fue posible usando el método *glScaled*, que tiene como argumentos (multiplicador\*x, multiplicador\*y, multiplicador\*z), de manera que la matriz original es multiplicada por la matriz de entrada, obteniendo como salida la imagen aumentada, en caso de que el valor sea mayor que 1, o la imagen disminuida en caso de que el valor sea menor que 1.

*Módulo de Ejecución:*

El mismo se encarga de imprimir en pantalla la imagen de salida, a partir de los parámetros de entrada del módulo de manipulación.

Para ello se utiliza como se ha mencionado las librerías de openGL, SDL y GUCVIEW, los cuales se explican como sigue:

*v4l2core\_get\_decoded\_frame():*

Este método se encarga de limpiar los buffers de v4l2.

*glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT):*

El mismo permite establecer la ventana de la imagen a través de valores seleccionados previamente. Para nuestro caso, se limpia el buffer del color o el buffer de profundidad, según corresponda.

*glBindTexture(GL\_TEXTURE\_2D, tex[0]):*

Este método se utiliza para especificar la textura que será ligada a la imagen de salida. En este caso, se utiliza la textura de 2D

*glBegin(GL\_QUADS) y glEnds():*

Se encarga de especificar las primitivas que serán creadas desde los vértices presentados entre *glBegin* y *glEnd*. Para este caso se utiliza *GL\_QUADS*, el cual permite manejar cada grupo de cuatro vértices como un cuadrilátero independiente.

*GlVertex2f():*

Especifica las coordenadas de los vértices x,y.

*glTexCoord2f():*

El mismo especifica las texturas coordinadas.

De esta manera la imagen es desplegada en pantalla con la resolución deseada, con los argumentos de entrada aplicados.

*v4l2core\_release\_frame()*

Se encarga de obtener el siguiente frame del vídeo para ser decodificado y manipulado nuevamente con los métodos descritos anteriormente.

# Resultados

En esta sección se explicará el resultado de la aplicación diseñada para cumplir con la solución del problema.

El programa tiene los siguientes argumentos de entrada por línea de comando:

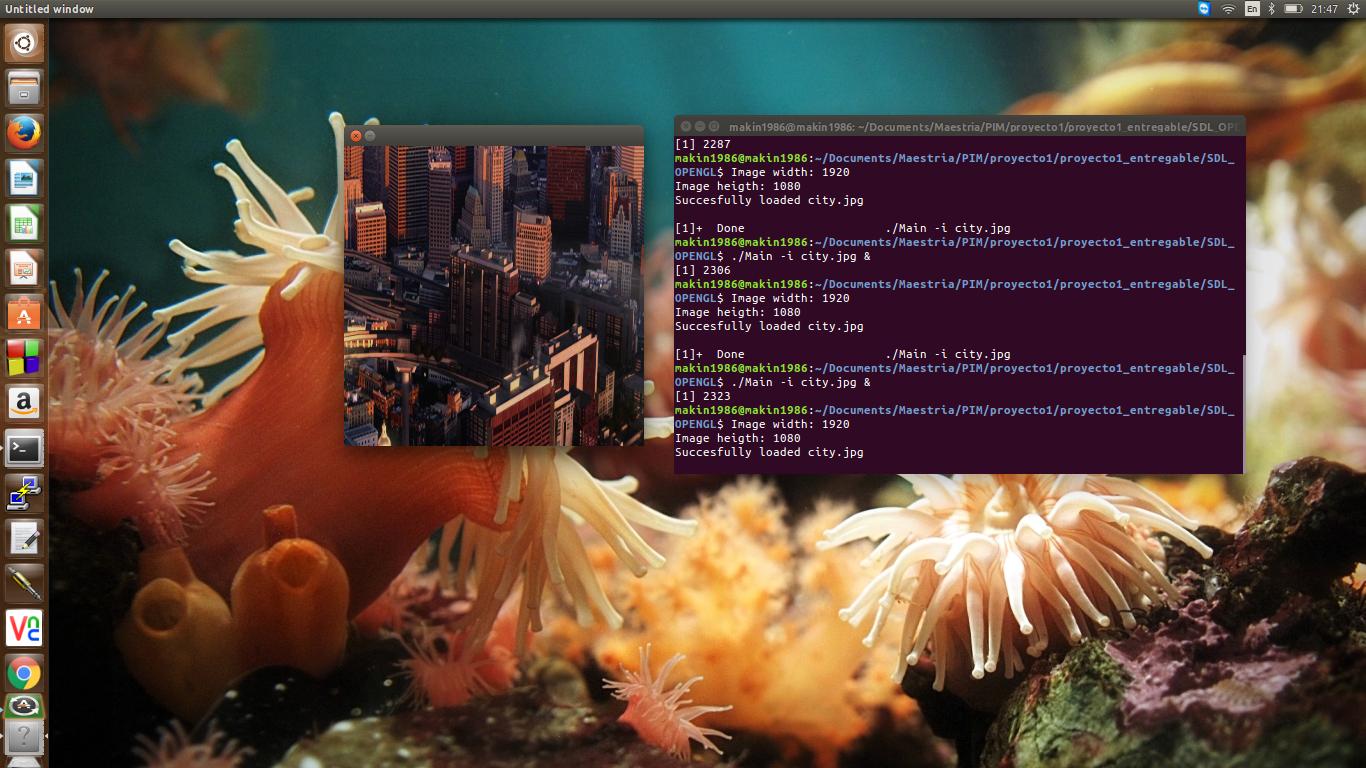
*-v:* Indica que se desea imprimir en pantalla los mensajes de la aplicación.

*-i:* Utilizado para indicar la ubicación de la imagen original que será cargada.

*-h:* Imprime en pantalla la ayuda del programa, donde se muestra cada uno de los argumentos de entrada para la línea de comando, y también se explica cada uno de los argumentos para manipular la imagen en tiempo real a través del teclado.

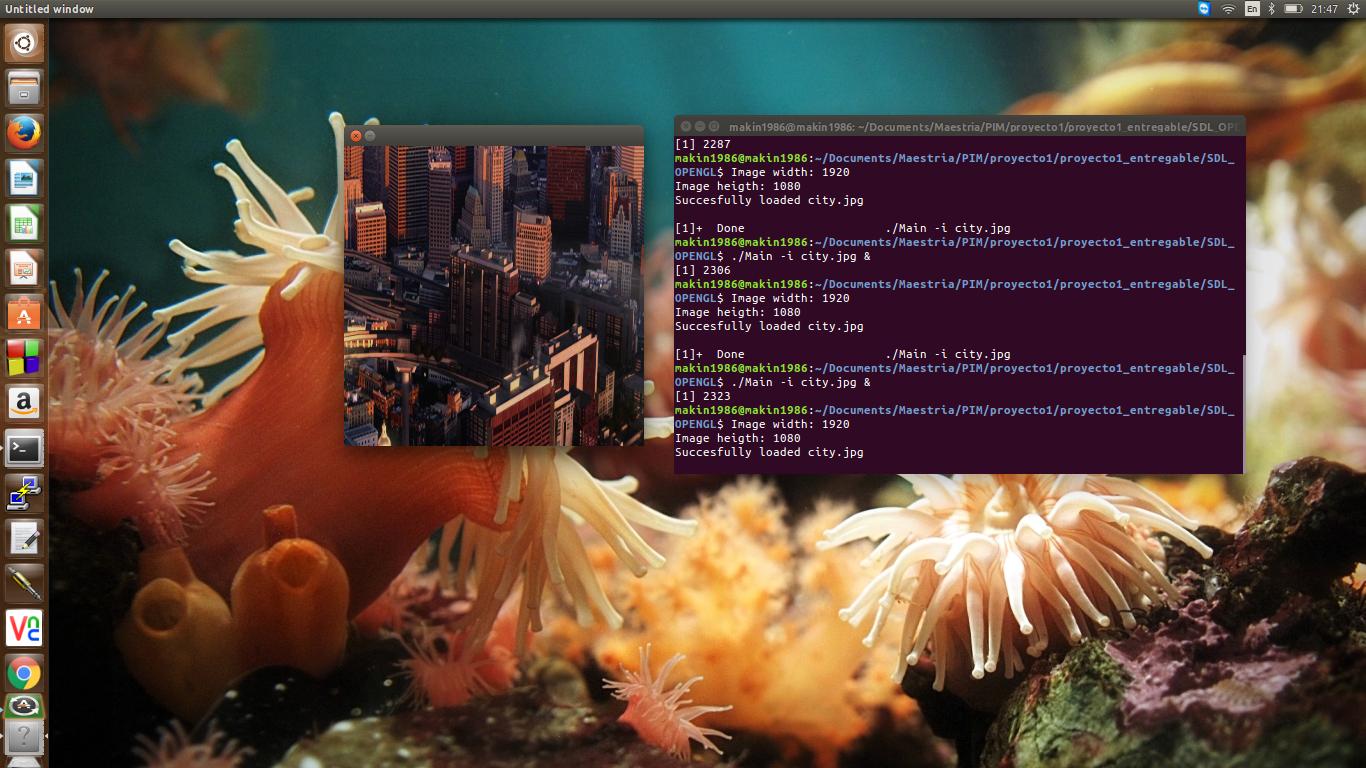
Seguidamente se muestra un ejemplo de la línea de comando para ejecutar el programa, así como también se muestra en la figura 2 la imagen original y la figura 3 la imagen de salida.

*./Main -i city.jpg*



**Figura 2.** Imagen original en resolución HD.

Como resultado se cargará la imagen *city.jpg,* que como se aprecia en la figura anterior, tiene una resolución de HD, pero a pesar de ello, la imagen de salida mostrada por la aplicación diseñada es de solamente 300x300 tal y como se muestra en la siguiente figura.

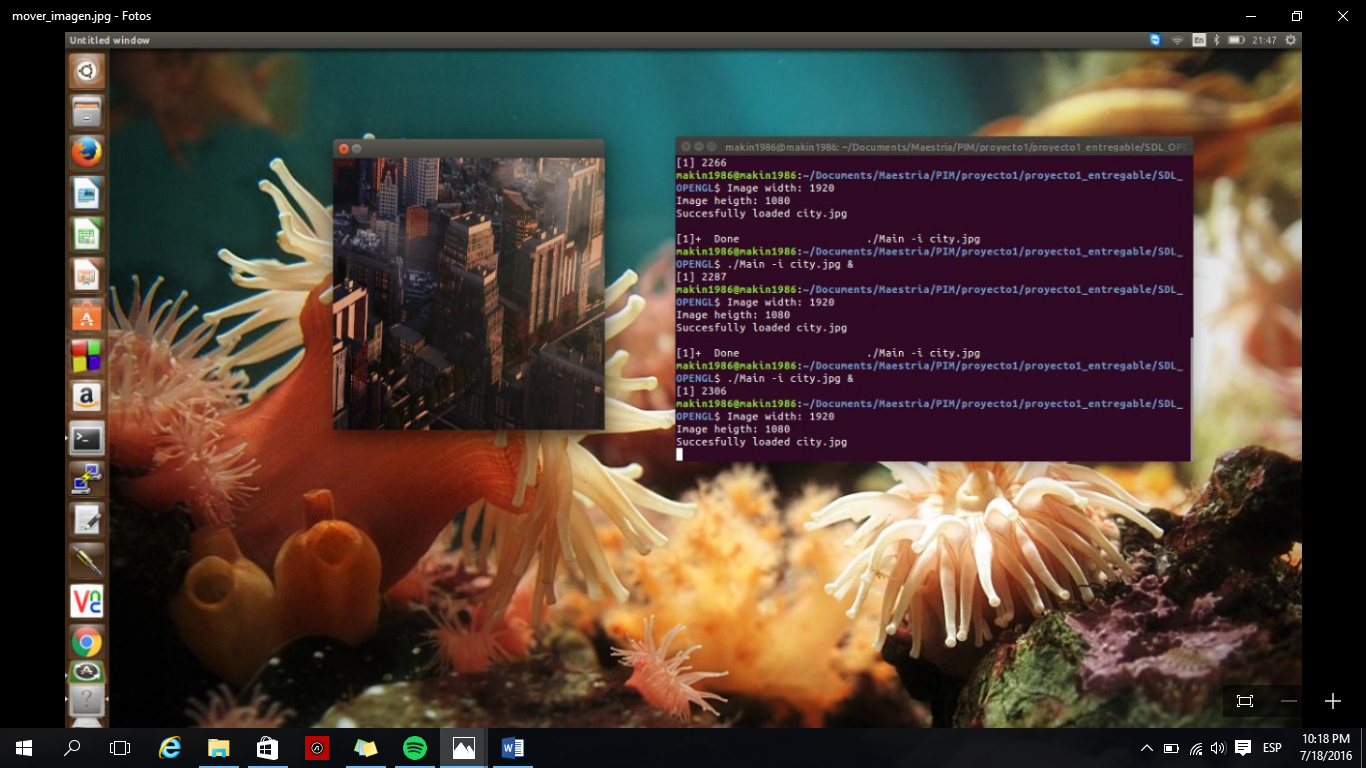


**Figura 3.** Imagen de salida con resolución 300x300.

Como se mencionó, la aplicación permite la manipulación de la imagen en tiempo real, de manera que la misma puede ser movida panorámicamente hacia la derecha, izquierda, arriba o abajo, también puede ser ampliada o disminuida y también puede ser rotada.

*Movimiento Panorámica:*

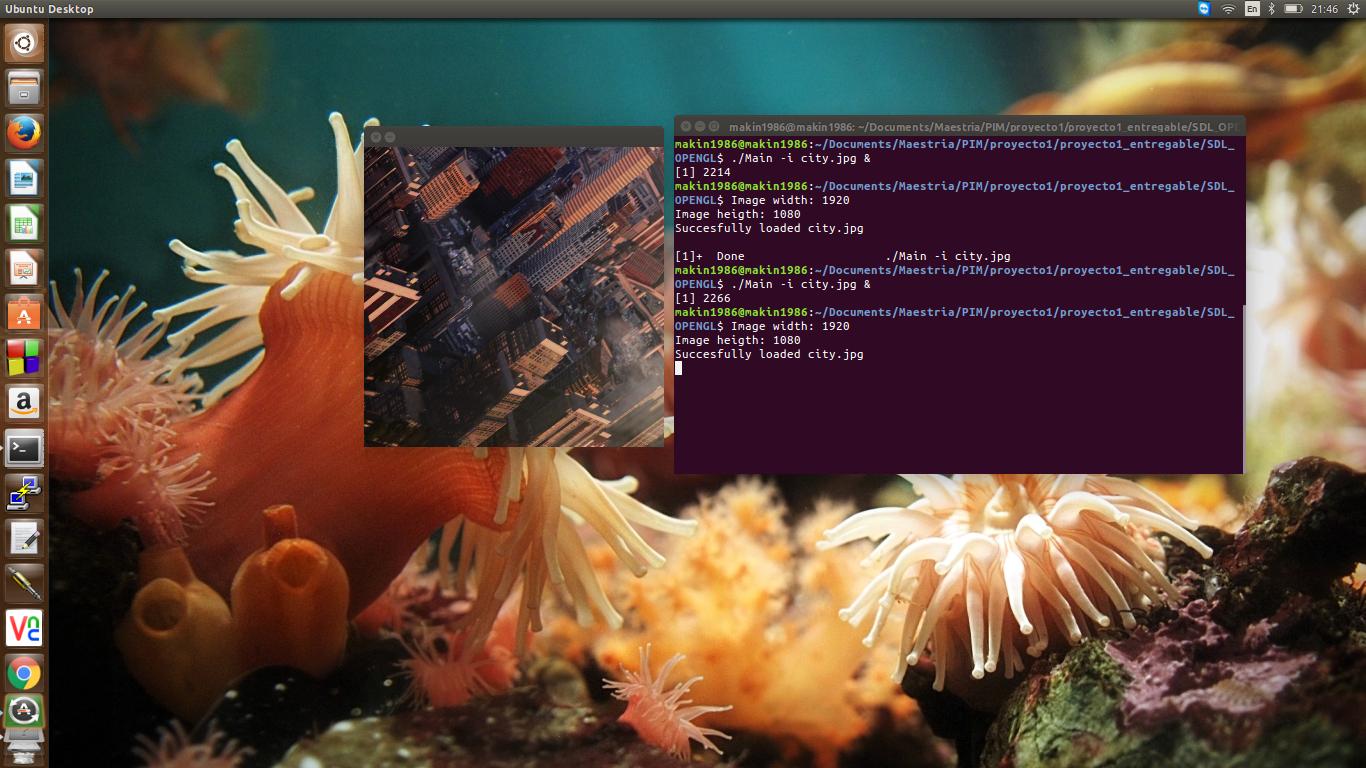
El mismo ocurre utilizando las teclas de derecha, izquierda, arriba o abajo. La siguiente figura muestra como la figura es movida hacia otra posición diferente a la que se aprecia en la imagen de salida inicial en la figura 3.



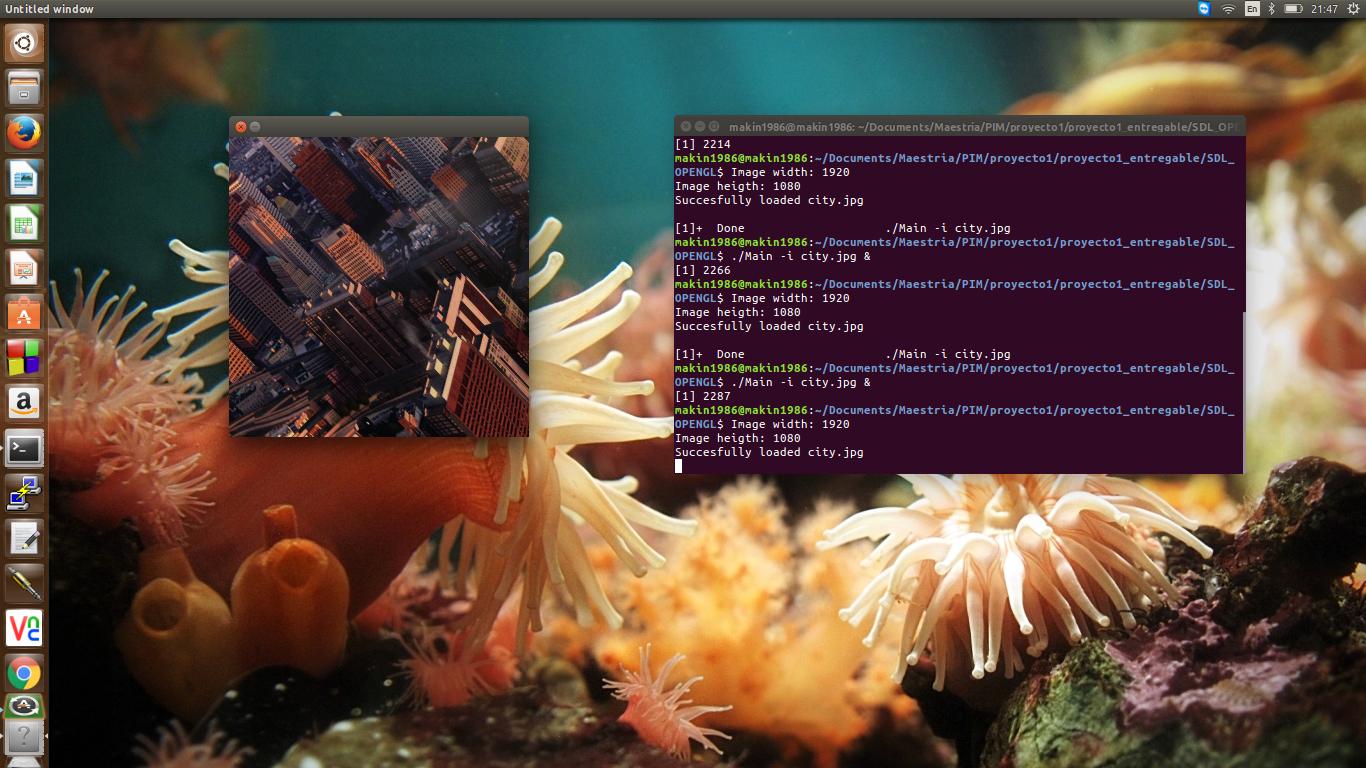
**Figura 4.** Imagen desplazada panorámicamente.

*Rotación hacia la derecha e izquierda:*

La rotación de la imagen se puede realizar utilizando las teclas *q* y *e*, de manera que la imagen puede tener una rotación hacia la derecha o hacia la izquierda. La figura 5 y 6 muestra la imagen con rotación hacia la derecha y con rotación hacia la izquierda respectivamente.



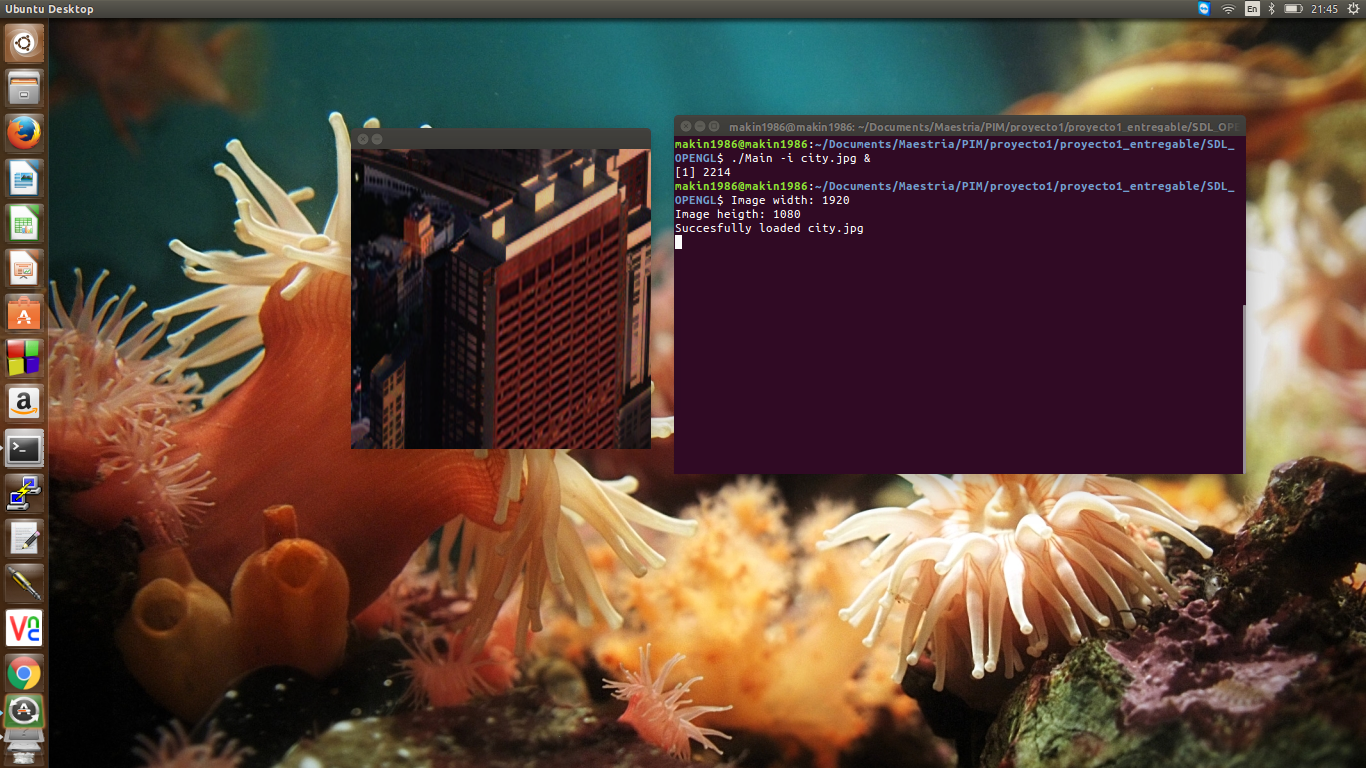
**Figura 5.** Imagen desplazada hacia la derecha utilizando la tecla q.

**Figura 6.** Imagen desplazada hacia la izquierda utilizando la tecla e.

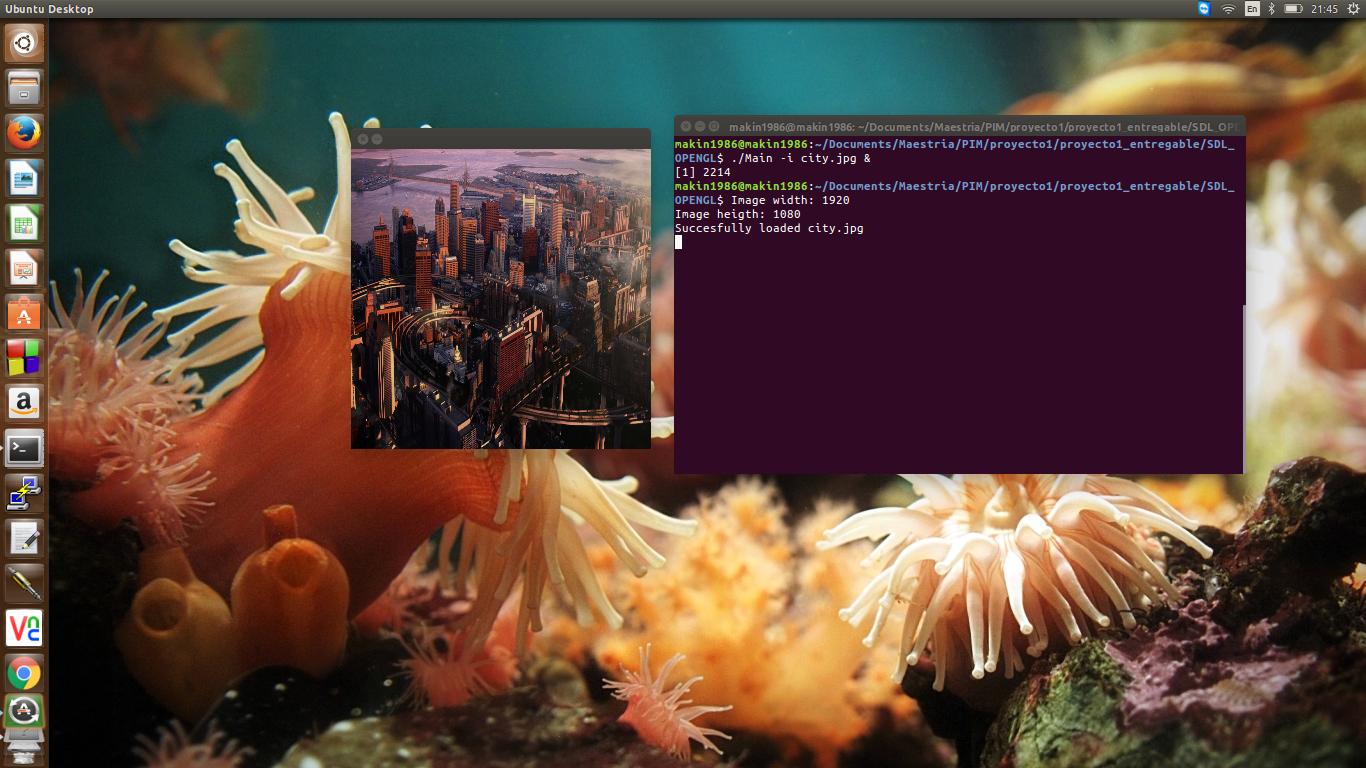
*Imagen ampliada y disminuida:*

La imagen puede ser ampliada utilizando la tecla *i* y puede ser disminuida utilizando la tecla *o*. De esta manera, se puede apreciar con más detalle algún específico de la imagen que se está analizando.

La figura 7 muestra la imagen ampliada y la figura 8 muestra la imagen disminuida.



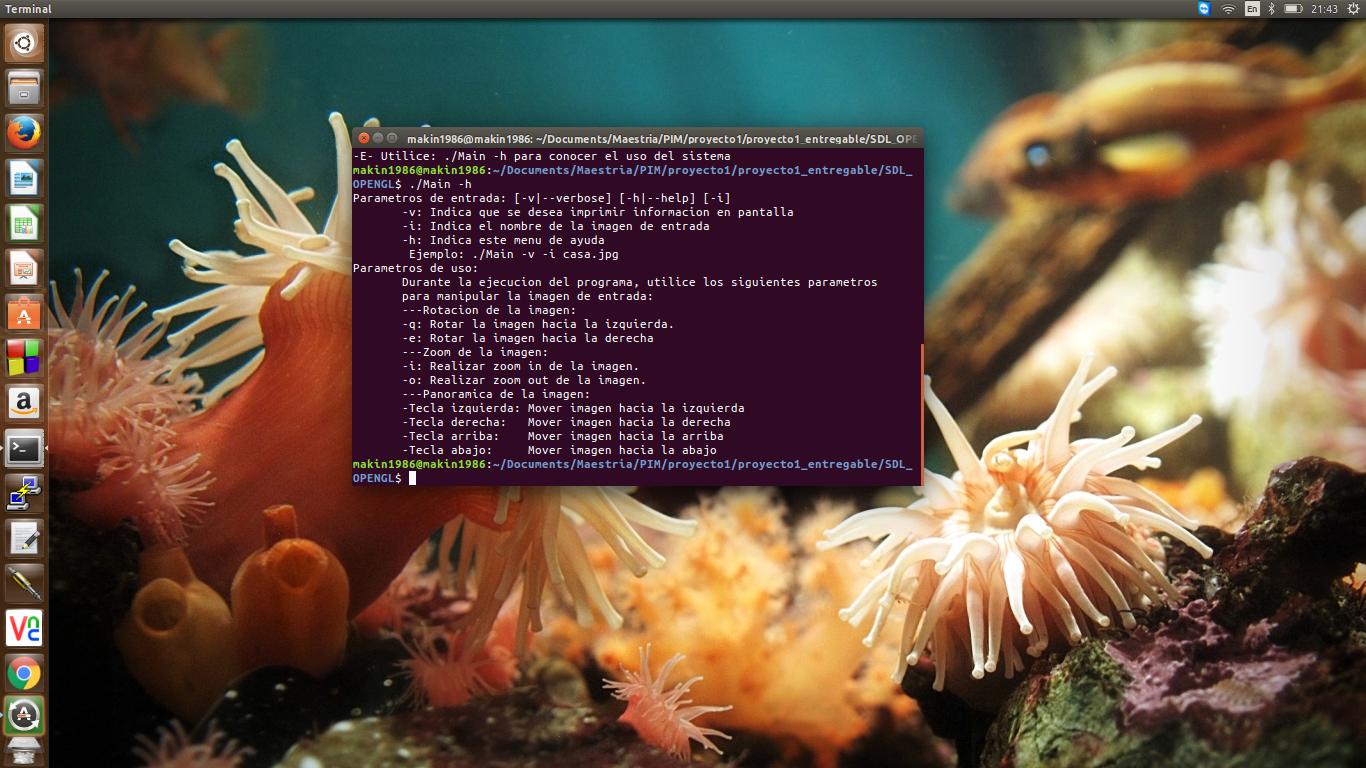
**Figura 7.** Imagen ampliada utilizando la tecla i.



**Figura 8.** Imagen disminuida utilizando la tecla o.

*Ayuda del programa:*

La aplicación desplegará un cuadro de ayuda que explica los argumentos que pueden ser utilizados en la línea de comando, y además muestra los argumentos de entrada que pueden ser utilizados en tiempo real para el análisis de la imagen. La figura 9 muestra el cuadro de ayuda desplegado por el programa.



**Figura 9.** Cuadro de ayuda desplegado por el programa.

# Conclusiones

Conclusiones

# Apéndices

Apéndices necesarios

# Referencias

Referencias.