Manual de usuario del Paquete “imageServer”.

Dr. Arturo Espinosa Romero
Dr. Carlos Brito Loeza
Dr. Ricardo Legarda Saenz

0.1. Introducción.

El objetivo de este paquete es el de proporcionar una herramienta que permita la distribución de imágenes capturadas desde una cámara a múltiples usuarios que se encuentran distribuidos en lugares diferentes, a través de internet. Las imágenes transmitidas pueden sufrir previa a su transmisión transformaciones lineales de rectificación y enmascaramiento, así como proveer mecanismos que faciliten el el análisis temporal de las imágenes, esto es considerar transformaciones que consideren el flujo de imágenes y no sólo imágenes aisladas.

El paquete imageServer es importante en en ámbitos de investigación aplicada así como la docencia, en donde es necesario que múltales actores, puedan acceder de manera simultanea a imágenes capturadas en un sitio. En particular este programa se aplica en dos actividades principalmente: en la captura y análisis de imágenes aéreas, y en la teleoperación de robots móviles en laboratorio.

La organización de este documento es como sigue: primero describimos las partes que constituyen el paquete computacional, a continuación se da una guía rápida de uso, para después presentar resultados obtenidos.

Este paquete fue desarrollado como parte de la segunda etapa del proyecto "Sistema Automático de Bajo Costo para la Captura y Análisis de Imágenes Aéreas"que se está realizando como parte de la red de Investigación Conjunta para Solución de Problemas en Adquisición, Procesamiento de Señales y Control Automáticos de Dispositivos de Sensado Remoto. apoyado por PRODEP.

0.2. Descripción.

El paquete se construyo utilizando estructuras de datos y métodos de la biblioteca openCV (Bradski, 2000) principalmente y esta compuesto por un conjunto de módulos que se detallan a continuación:

- **ImageServer:** Este modulo consiste en un conjunto de biblioteca de funciones que permiten construir un servidor capaz de atender múltiples clientes de manera concurrente, y atender solicitudes de éstos para enviar imágenes. El programa crea un *stream socket*, que se asocia a una dirección IP(v4). Cuando un cliente se conecta se crea una nueva hebra para atender dicho cliente, y permitirle acceder a imágenes capturadas.

- **Camera** Este módulo captura imágenes de un dispositivos de captura y las almacena en una *buffer* (cola circular). Provee mecanismos de sincronización para controlar el acceso a los elementos de la cola.

- **imgServer** Es el programa donde los diferentes componente necesarios para construir el servidor de imágenes se combinan. El programa recibe como parámetros las dirección IP(v4) y número de puerto que se va a asociar al servidor, y un archivo que contiene una mascara que se aplicara a las imágenes capturadas. Cada imagen capturada por el usuario se le somete a dos transformaciones geométricas. La

primera sirve para rectificar la imagen, en caso necesario, y la segunda para ajustar el tamaño de la máscara al tamaño de la imagen capturada. El modelo de rectificación se calcula a partir de información que provee el usuario, seleccionando la región planar de la escena a observar.

- **Client** El cliente consiste de dos principales miembros. Una clase que provee de métodos que facilitan la conexión con el servidor de imágenes, así como la descarga de imágenes de éste, y dos programas de ejemplo una escrito en C++, y el otro un *script* de python que sirven como guía para desarrollar programas mas complejos, y el uso del servicio en otros ámbitos.

0.3. Guía de uso.

El código esta preparado para ser compilado en una computadora que tenga como sistema operativo alguna variante de Unix (*e.g.* Linux, MacOS), aunque es posible también compilarlo y utilizarlo en otras plataformas.

La interacción con el programa se hace invocándolo desde una emulador de terminal. Primero que hay que ubicarse en el directorio en donde está el código y compilarlo utilizando el programa make[1].

```
$ make
```

Un vez que se ha compilado el código se puede invocar como sigue:

```
$ ./imgServer CamId IPAddres Puerto ArchivoMascara
```

El primero parámetro es el identificador de cámara que se va a utilizar (un numero entero, el segundo parámetro es una direccion IP(v4), el tercer parámetro es el numero de puerto y por último el último parámetro es el archivo que contiene la imagen que

[1] se asume que en el sistema está instalado las bibliotecas de desarrollo de **openCV**

se va a usar como máscara. La máscara opera como sigue: en aquellos pixeles de la máscara que son diferentes a RGB[255,255,255] (color blanco), el pixel correspondiente de la imagen se sustituye por el valor de la máscara. De caso contrario, el valor original del pixel, se preserva.

Una vez que se invoca, el programa abre una ventana en donde se muestra la escena que esta siendo capturada por la cámara. Aqui el usuario puede seleccionar las cuatro esquinas de un paralelogramo contenido en algun plano de la imagen, con el fin de calcular la rectificación, en dado caso que ésto no sea necesario, solo hay que seleccionar las cuatro esquinas de la imagen, comenzado desde la esquina superior izquierda procediendo a favor de las manecillas del reloj.

Una vez que se ha hecho, en la consola desde donde se invocó el programa se muestra las matrices de transformación calculadas, y el servidor permanece en ejecución aceptando conexiones de los clientes del sistema. En la figura1 se muestra una captura de dicha consola así como la imagen en donde se muestran las cuatro esquinas capturadas.

Una vez que se ejecuta el servidor, desde cualquier otra computadora que pueda acceder al servidor puede ejecutarse un archivo cliente que obtenga las imagenes. En nuestro caso utilizamos el programa `testClient`:

```
$ ./testClient IPAddres Puerto
```

El primero parámetro es una direccion IP(v4), y el segundo parámetro es el numero de puerto. Al ejecutarse, el programa abre una ventana y muestra las imagenes que recibe del servidor. Ejemplos de las imagenes que recibe se muestran en la figura 2. Dichos ejemplos muestra un ejemplo del sistema en una operación de teleoperación de robots móviles.

```
/bin/bash
/bin/bash  /bin/bash  /bin/bash  /bin/bash  /bin/bash  /bin/bash  /bin/bash
/bin/bash 80x24
opengl support available
Se capturaron los siguientes puntos:
(28, 39)
(315, 98)
(329, 434)
(35, 461)
Calculamos la Hrv como:
[1.754176571658634, -0.02909771564362738, -47.98213309634026;
  -0.2389369072273949, 1.16228631142817, -38.63893274333187;
  -0.000712615408229121, 9.853890467331724e-05, 1]
[640, 480]
Calculamos la Hmv como:
[0.9984375000000023, 2.063068174683411e-15, -4.5130565951012 61e-13;
  1.882827186804222e-16, 0.9979166666666703, -5.160195187814409e-14;
  2.718224680690106e-18, 6.4942661266865e-18, 1]
Se acaba de hacer una nueva conexión: 1
Se acaba de hacer una nueva conexión: 2
Se acaba de hacer una nueva conexión: 3
Se acaba de hacer una nueva conexión: 4
Recibimos QUIT
Enviamos BYE
Cliente con id: 22 Teminó
Se acaba de hacer una nueva conexión: 5
```

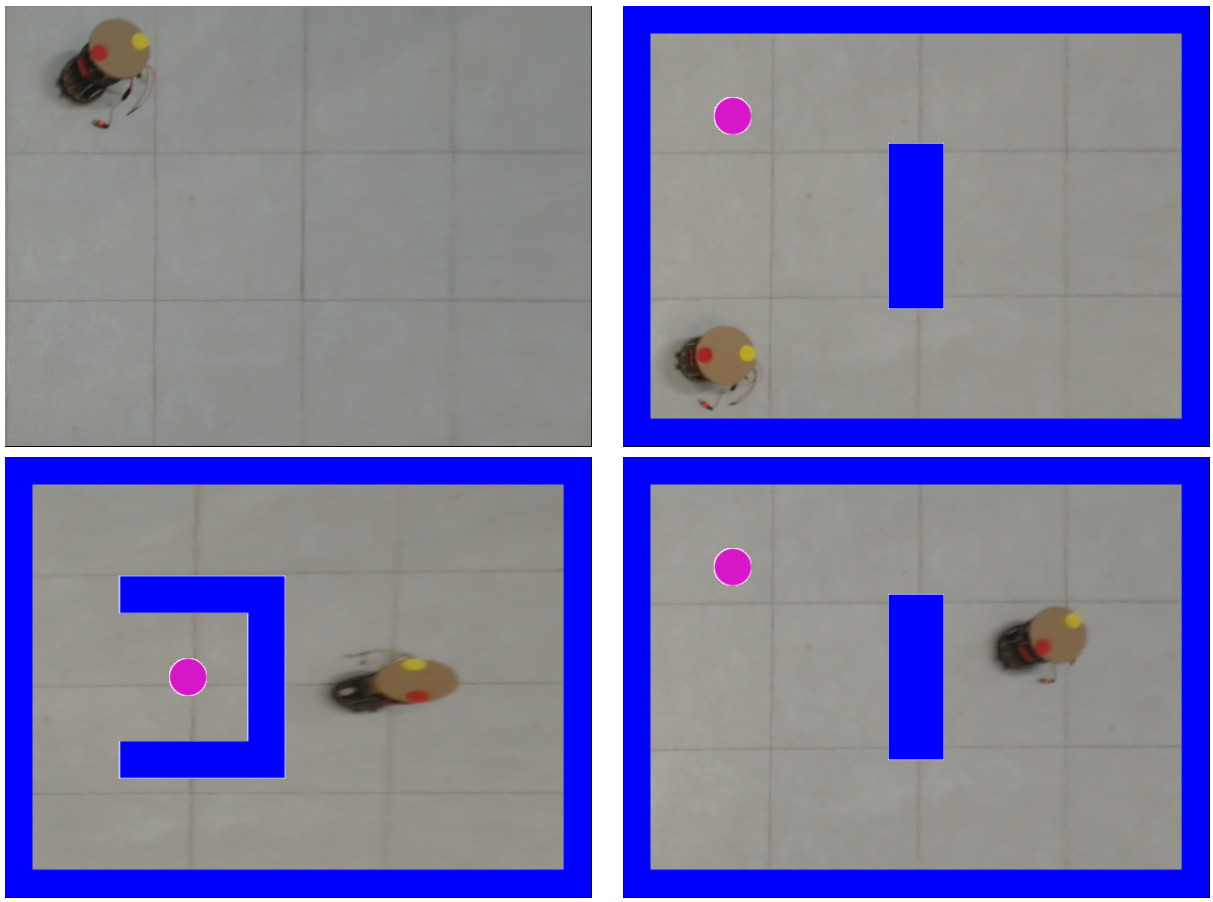

Figura 2: Imagenes mostradas por el programa testClient. En la esquina superior izquiera se muestra una imagen capturada sin utilizar ninguna máscara, y las tres siguientes imagenes muestran imágenes con algun tipo de máscara añadidio.

Bibliografía

Bradski, G. (2000). *Dr. Dobb's Journal of Software Tools*.