Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores

(Licentiate Degree Program in Computer Engineering)

Curso: CE-5201 Análisis y procesamiento de imágenes

(Course: CE-5201 Analysis and image processing)

Proyecto 1

Realizado por:

Made by:

Wilberth Varela Guillén

Profesor:

(Professor)

Ing. Pablo Alvarado Moya

Fecha: Cartago, Marzo 13, 2018

(Date: March 13th ,2018)

1. Estructura del código

1.1 Función take_Photos

Esta función tiene como trabajo de tomar las fotos necesarias para la validación y funcionamiento de la calibración de la camara. Primero se presenta un menú el cual le pide que ingrese la carpeta destino donde se van a guardar las imágenes. La estructura de esta dirección tiene que ser como la que se indica de ejemplo cuando solicita la dirección. Luego de ingresar esta función, se abre una ventana donde se toman las fotos, para tomar una foto se presiona la tecla “s”, si se desea salir se presiona la tecla “Esc”, igual se van a tomar las fotos hasta un máximo de 14 fotos, si no se presiona la tecla de “Esc”, estas 14 fotos son la cantidad minima para calibrar la cámara.

1.2 Función calcChessboardCorners

Esta función se encarga de calcular las esquinas del tablero, los parámetros que entran son: el tamaño del tablero de ajedrez, el tamaño de los cuadros en milímetros, el vector que va a contener las esquinas. Se va hacer una “for” con el fin de recorrer el alto y el ancho del tablero contando las esquinas internas, así conforme se va recorriendo el tamaño se van ingresando las posiciones donde están cada cuadro, con su respectivas coordenadas.

1.3 Función saveparams

El objetivo de esta función es poder guardar los parámetros generados de la exactitud de calibración, la matriz de la cámara, los coeficientes de distorsión, el vector de rotación, vectores de rotación y el vectores de traslación, los parámetros que entran la dirección donde se va a guardar el archivo “filename”, la variable de la matriz de la cámara “camaraMatrix”, ademas de la variable de los coeficientes de distorsión “distCoeffs”, los vectores de rotación “rvecs”, los vectores de traslación “tvecs” y su exactitud de calibración. Como los vectores de rotación y traslación se

hace se tienen que validar si no estan vacios, con el fin de no agregar basura al archivo, si es así entonces se recorren los valores de los vectores con su respectivos parámetros, ya que son vectores entonces tienen varios valores, por ende se tienen que recorrer hasta que termine. y los va agregando al archivo conforme salgan los tres valores de cada posición del vector, una vez terminado se cierra el archivo con los datos codificados de la calibración de la cámara.

1.4 Función calibrate_Display_Result

En esta función se hace los trabajos más pesados, primeramente se declaran la variables necesarias para poder calibrar la cámara, entre las variables más importantes son las esquinas en 3D “corners3D”, las esquinas en 2D “corners2D”, las coordenadas en 3D “coord3D” y las coordenadas en 2D “coord2D”, luego acá se pide la dirección de la carpeta donde están ubicadas las fotos para la calibración de la cámara. Esta dirección tiene una estructura ya definida, en el menú de esta dirección se brinda un ejemplo de cómo se tiene que ingresar esta dirección. Luego se calculan las esquinas del tablero con la función 1.2, luego se hace un “for” de 14 elementos, para poder abrir cada imagen y aplicarle sus respectivas operaciones.
Conforme se abre una imagen se imprime en consola su dirección, una vez la imagen abierta, con la función findChessboardCorners(), se encuentra cada esquina del tablero de ajedrez, con la imagen en gris, que se cargó, el tamaño del tablero, las esquinas en 2D y además de esto una serie de constantes pertenecientes a la función que se ocupa para poder encontrar las esquinas, estas constantes pueden ser:

- CALIB_CB_ADAPTIVE_THRESH: Use el umbral adaptable para convertir la imagen a blanco y negro, en lugar de un nivel de umbral fijo.
- CALIB_CB_NORMALIZE_IMAGE: Normalice la gama de la imagen con equalizeHist antes de aplicar el umbral fijo o adaptativo.
- CALIB_CB_FAST_CHECK: Ejecute una comprobación rápida de la imagen que busca las esquinas del tablero de ajedrez, y acceda a la llamada si no encuentra ninguna. Esto puede acelerar drásticamente la llamada en la condición degenerada cuando no se observa el tablero de ajedrez.

Esta función retorna un true si encuentra las esquinas, o false si no las encuentra, en caso de true entra y aplica la función de cornerSubPix(), la cual se encarga de marcar los puntos donde sacó de la función anterior, luego de ello utiliza la función de drawChessboardCorners() con el fin de dibujar los puntos donde se encuentran en el tablero, por último se agrega los puntos los vectores respectivos, además de presentar cada imagen con sus puntos.

Luego se declaran las variables de la matriz de la cámara, los coeficientes de distorsión entre las otras variables a guardar. Se aplica la función de opencv de calibrateCamera(), la cual le ingresan las coordenadas de 2D y 3D, el tamaño de la imagen, la matriz de la cámara, y las otras variables como los coeficientes de distorsión, los vectores de rotación y traslación, además de una serie de parámetros de opecv que ocupa la función para su respectivo funcionamiento. Entre estos parámetros se pueden tener, se pueden tener más:

- CALIB_FIX_PRINCIPAL_POINT: El punto principal no se cambia durante la optimización global.
- CALIB_FIX_ASPECT_RATIO: Las funciones solo consideran fy como un parámetro libre. La relación fx / fy permanece igual que en la cámara de entradaMatriz
- CALIB_ZERO_TANGENT_DIST: Los coeficientes de distorsión tangencial (p1, p2) se establecen en ceros y permanecen en cero.

Luego de esto se imprimen las variables principales de la calibración y se llama a la función 1.3 qué es para guardar todos estos parámetros.

A continuación lo que sigue es poder presentar un punto con un eje coordenado con las direcciones de “X”, “Y” y “Z”. Primero se declaran la variables necesarias para esta sección, las cuales están los vectores de la biblioteca “Mat”, asi como las variables de opencv para abrir la cámara y poder captar video en vivo. luegos las constantes string que se van a presentar en la las coordenadas del tablero cuando se proyecte un tablero a la cámara.

Luego de ello se declaran puntos en 3D “points3D” con el fin de saber las ubicaciones de los puntos donde se va a finalizar el vector, del eje coordenado. luego entra en un ciclo infinito, para seguir vuelve a encontrar las esquinas, con los mismo parámetros que se habían hecho en las sección anterior. Luego de ello

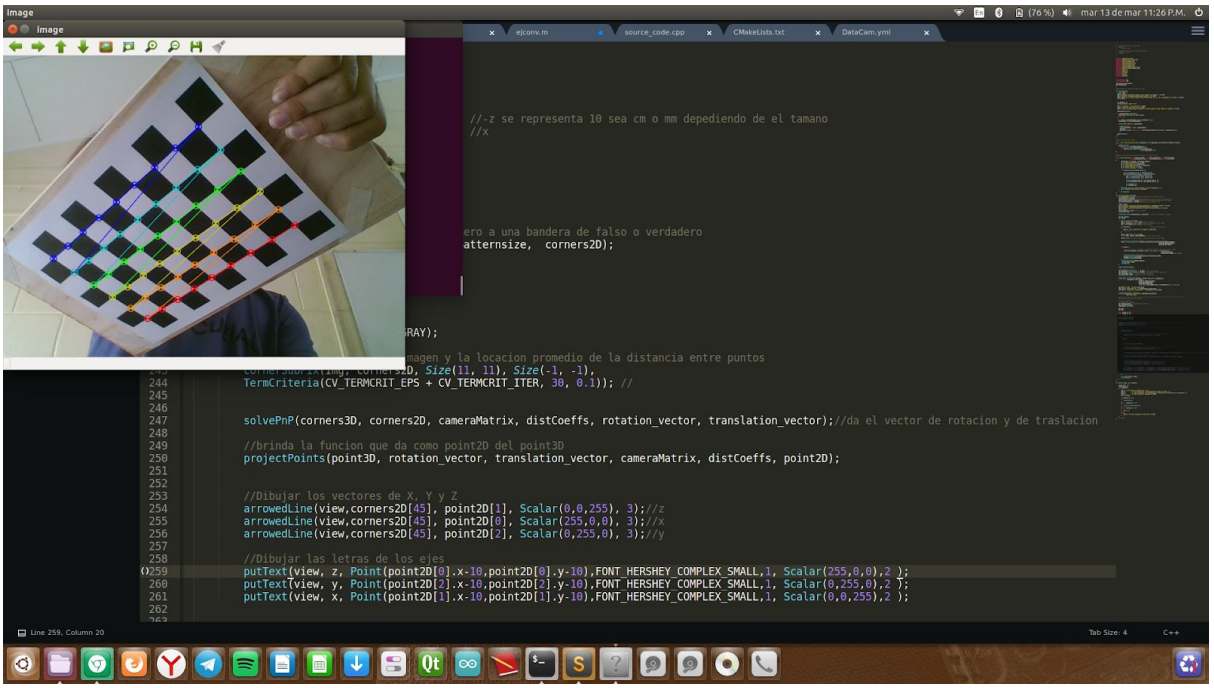
vuelve a marcar los puntos como la sección anterior mencionada y con los mismos parámetros. Para continuar con la función solvePnP() calcula los vectores de rotación y traslación según los diferentes puntos del tablero, para cuando el tablero se mueva pueda seguir el vector según el punto definido y no se pierda el punto ni el eje coordenado dibujado. Con la función projectPoints() proyecta los puntos del tablero con el fin de poderlos ubicar y no perder su posición sin importar si el tablero se mueve con los diferentes parámetros de la matriz de la cámara, los coeficientes y los vectores calculados anteriormente. Con la función arrowedLine() dibujo una fecha desde el punto que esto definiendo de origen entre todos los puntos del tablero y el punto que esto ubicando en con los puntos en 3D “points3D”, además con la función putText(), dibuja la letras antes declaradas a su respectivo eje con el fin de que se pueda identificar cada eje. Por último se proyecta el video con el resultado si se presenta el tablero definido.

1.5 Función Main

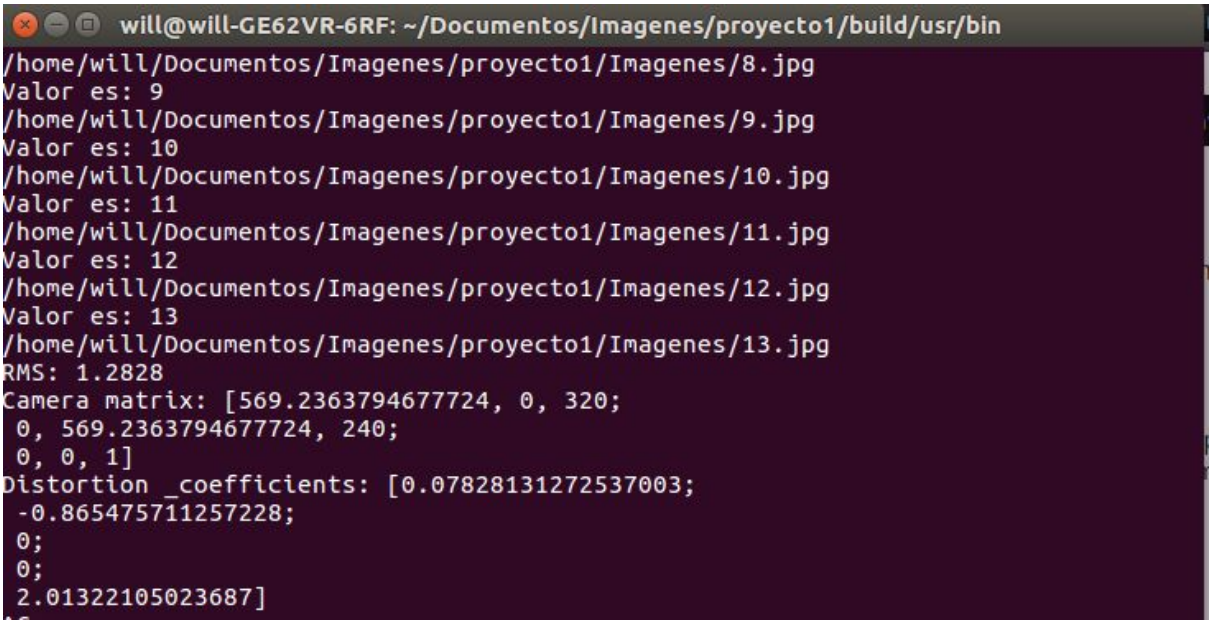
En el main solo se establece un menú pequeño con el fin de poder escoger entre las funcionalidades que son el poder tomar una foto, o iniciar la calibración y proyección del resultado de la calibración.

Para poder elegir tomar las fotos se escribe “-t”, para poder elegir la calibración y proyección de resultados se escribe en consola “‘-c” o para terminar el programa se escribe “-e”.

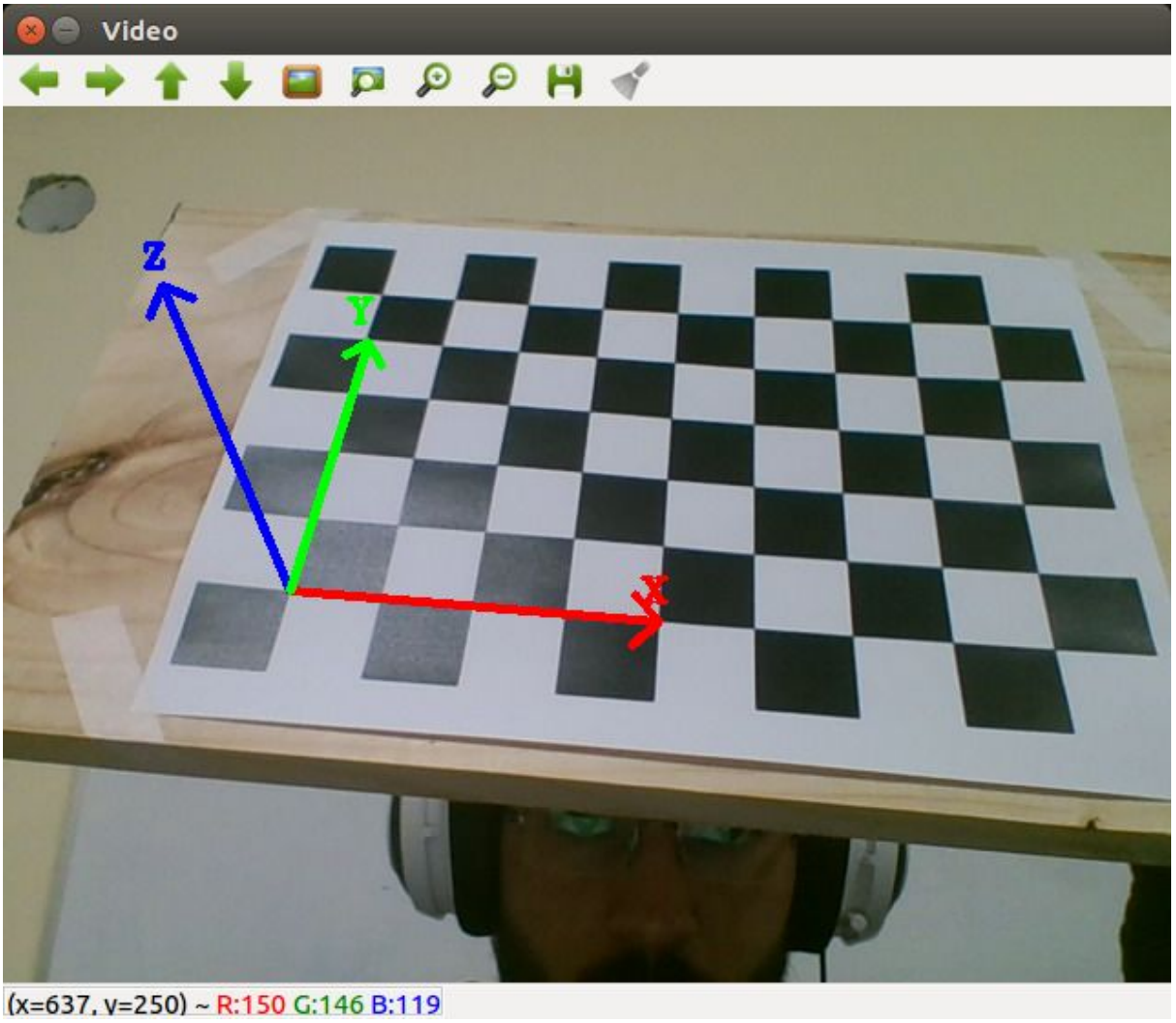
2. Resultados.

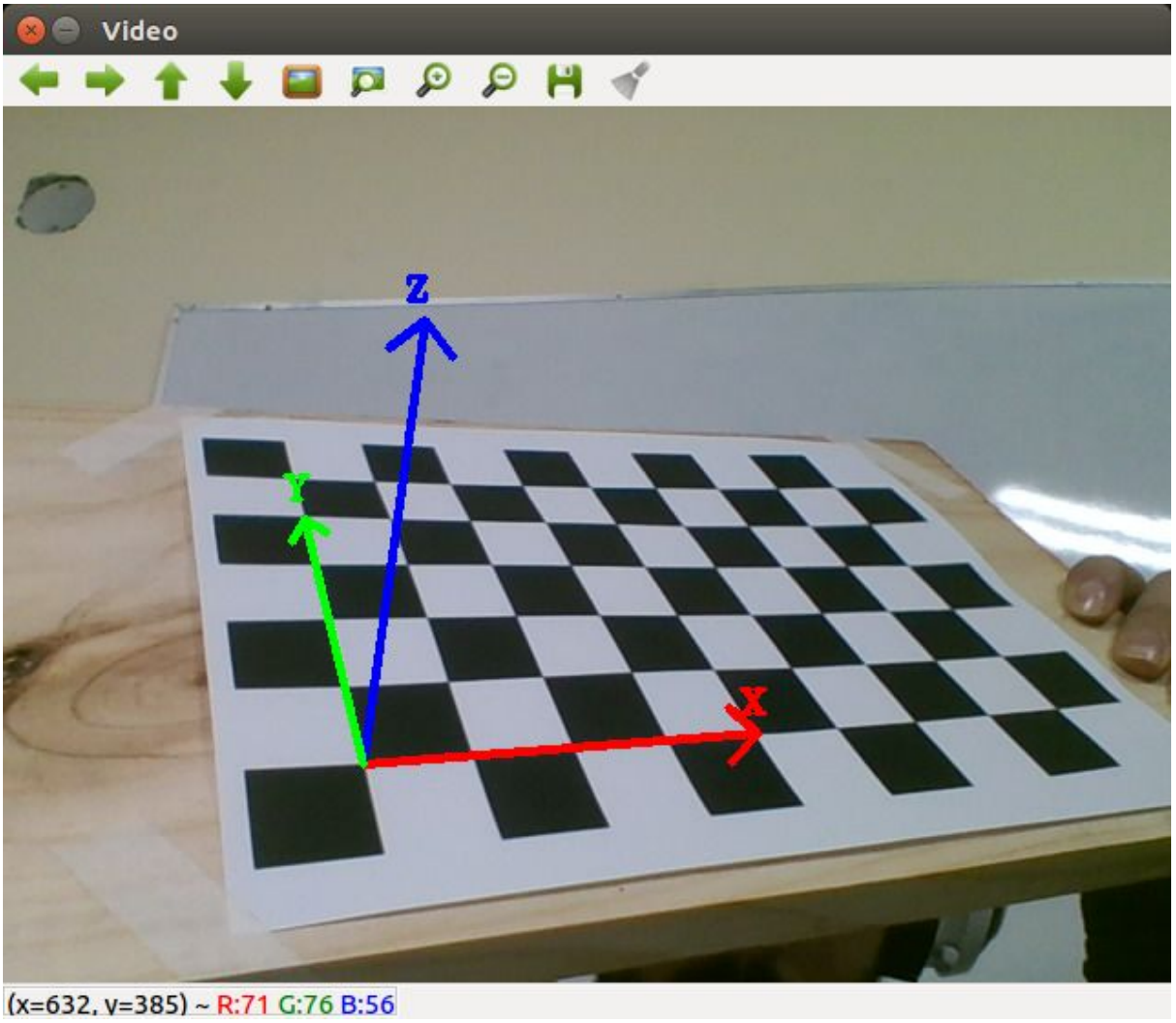


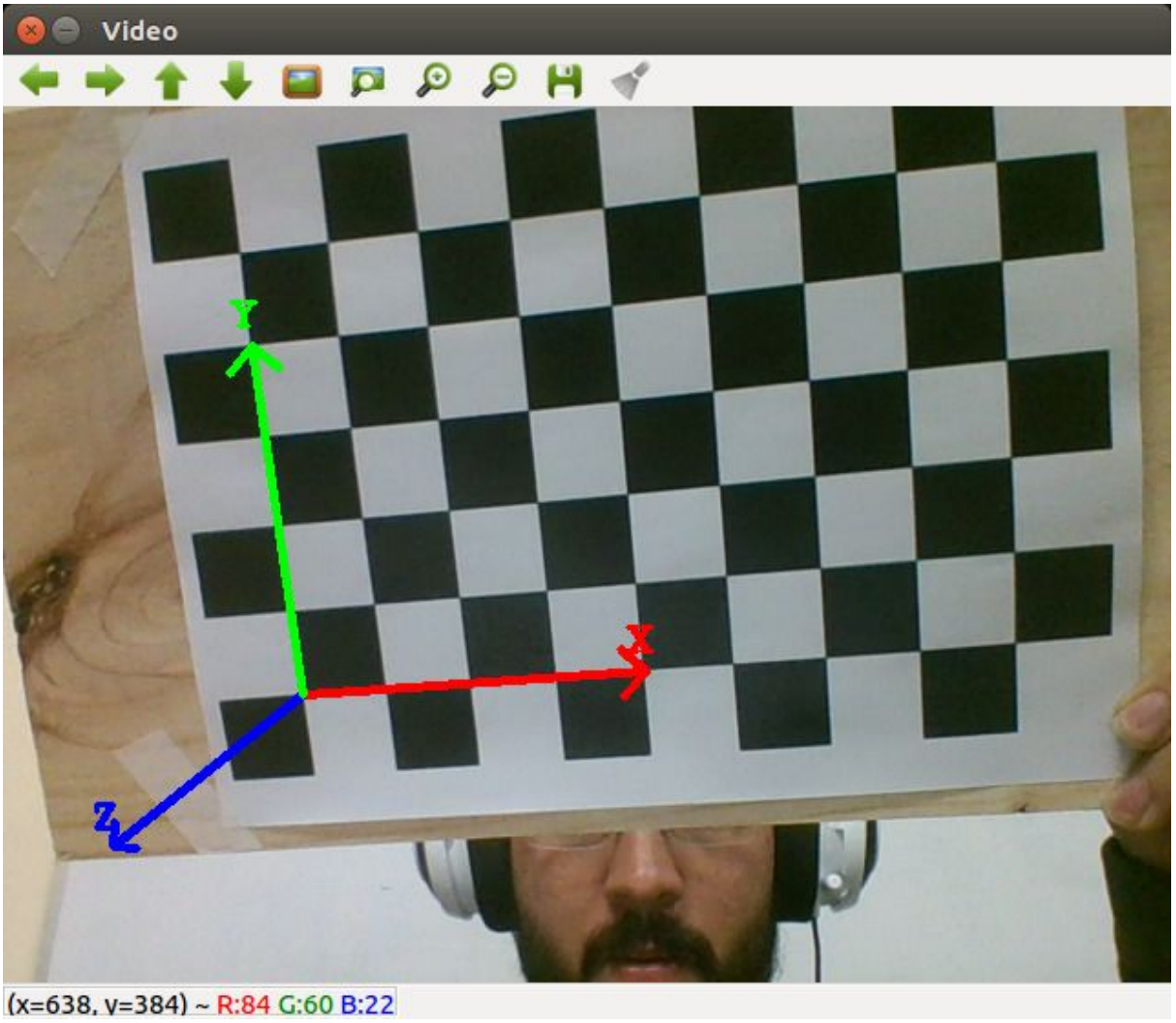
2.1 La figura donde se muestra como se van cargando las imágenes para su respectiva calibración.



2.2 Aca se muestra el resultado.


Video
Z
Y
X
(x=637, y=250) ~ R:150 G:146 B:119


Video
z
y
x
(x=632, y=385) ~ R:71 G:76 B:56



2.3 diferentes Ángulos del resultado con los vectores y enseñando el tablero

3.Ejecución del programa

- Descargar el .tar
- Se descomprime el .tar en la root de sistema, posible dirección: "/home/USUARIO/"

  Nota: donde el USUARIO se cambia. por el usuario del computador.
- Entra a la carpeta "/proyecto1/build/"
- Dentro de este punto, abre una ventana de terminal.
- Ejecuta el comando:

cmake ../ -DCMAKE_INSTALL_PREFIX:PATH=/home/USUARIO/proyecto1/build/usr

Nota: Donde USUARIO debe reemplazar el usuario del sistema.

- Construccion de programa: Para la construcción del programa (modo usuario) únicamente se debe ejecutar el comando

$ make

- Instalación: Para la instalación del programa se ejecuta

$ make install

Nota: El comando anterior instala el binario en el directorio usr/bin.

- Verificación: Para verificar el programa se requiere la ejecución del mismo.

$ cd usr/bin && ./proyecto1