UNIVERSIDAD DE MÁLAGA

Casos de prueba

Proyecto VIGIA

Autores:

Rafael Martínez González
Israel Rodríguez Martín
Jesús Manuel Rodríguez Sánchez

Versión: 3.0

Fecha: 24/01/2009

Tabla de contenido

1ª iteración 4

Conexión Mando-PC 4

2ª iteración 6

Aplicación que mueve la cámara 6

Aplicación de comunicación Cliente-Servidor 8

Aplicación CMC (Coordenadas Mando Consola) 9

LED's 12

Algoritmo de recorte 13

3ª iteración 17

Integración: enviar órdenes 17

4ª Iteración 19

Integración: envío órdenes / recepción frames 19

Integración y calibración del sistema 22

1ª iteración

Conexión Mando-PC

Conexión Mando-PC en Linux

- **Descripción del evento**

La prueba consistirá en lograr que el PC reconozca los movimientos del mando de la Wii (Wiimote), y que, a través de la librería de Linux que vamos a usar, nos aporte ciertos parámetros relativos a la posición del mando, la intensidad de la señal, etc.

Hardware necesario: equipo de pruebas.

Software necesario: librería Cwiid.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los movimientos horizontales, verticales y en profundidad del mando Wiimote.

- **Eventos a observar**

Los programas que vamos a usar nos mostrarán valores referentes a la posición del mando, además de otros parámetros. También podremos simular el funcionamiento del ratón, ya que uno de los programas incluidos en la librería Cwiid permite que el Wiimote haga de ratón.

- **Consecuencias en caso de fallo**

No podremos establecer una conexión mando-PC en Linux, por lo que no sabremos en qué posición se encuentra el usuario del sistema.

Conexión Mando-PC en Windows

- **Descripción del evento**

La prueba consistirá en lograr que el PC reconozca los movimientos del Wiimote, y que, a través de la aplicación de Windows que vamos a usar, nos aporte

ciertos parámetros relativos a la posición del mando, la intensidad de la señal, etc.

Hardware necesario: equipo de pruebas.

Software necesario: BlueSoleil y WiinRemote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los movimientos horizontales, verticales y en profundidad del mando de la Wii.

- **Eventos a observar**

La aplicación que vamos a usar nos mostrará parámetros referentes a la posición del Wiimote, además de los botones que pulsamos, la intensidad de la señal emitida...

- **Consecuencias en caso de fallo**

No podremos establecer una conexión mando-PC en Windows, por lo que no sabremos en qué posición se encuentra el usuario del sistema.

2ª iteración

Aplicación que mueve la cámara

Movimiento automático de la base de la cámara

- **Descripción del evento**

La prueba consistirá en decirle de antemano a la base de la cámara qué movimientos debe realizar. Éstos se ejecutarán uno trás otro con pequeños desfases entre ellos para poder observar los movimientos por separado.

Hardware necesario: base de la cámara y equipo de pruebas.

Software necesario: clase de prueba implementada en Java para controlar la base.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una serie de desplazamientos (x, y, z) a la que deberá responder adecuadamente.

- **Eventos a observar**

La base de la cámara se desplazará según los parámetros indicados anteriormente.

- **Consecuencias en caso de fallo**

No podremos programar una serie de movimientos previamente definidos, aunque, realmente, el objetivo es que responda a los movimientos del usuario, y no a acciones ya establecidas.

Movimiento de la base de la cámara ante una acción propuesta por el usuario

- **Descripción del evento**

El usuario le indica, a través de la consola de Java, a la base de la cámara un desplazamiento (x, y, z) al que deberá responder debidamente.

Hardware necesario: base de la cámara y equipo de pruebas.

Software necesario: clase de prueba implementada en Java para controlar la base.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los parámetros 'x', 'y' y 'z' que definen el movimiento que tiene que llevar a cabo la base.

- **Eventos a observar**

La base se moverá según le hemos indicado en la acción anterior.

- **Consecuencias en caso de fallo**

La base de la cámara no responderá ante un movimiento del usuario, por lo que el sistema no funcionaría.

Movimiento de la base de la cámara ante una serie de acciones del usuario

- **Descripción del evento**

El usuario le indica a la base de la cámara, a través de la consola de Java, una secuencia de desplazamientos (x, y, z) a la que deberá responder adecuadamente.

Hardware necesario: base de la cámara y equpo de pruebas.

Software necesario: clase de prueba implementada en Java para controlar la base.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una serie de ternas (x, y, z) que representan los distintos movimientos a realizar por la base.

- **Eventos a observar**

La base de la cámara, ante una acción indicada por el usuario en la consola de Java, se moverá atendiendo a los valores especificados por el usuario.

- **Consecuencias en caso de fallo**

La plataforma no podrá responder ante una serie de desplazamientos del usuario, por lo cual no podremos simular por pantalla lo que vería dicho usuario si estuviera donde está la cámara.

Aplicación de comunicación Cliente-Servidor

Establecimiento de una conexión Cliente-Servidor con envío de frames

- **Descripción del evento**

Se crea un canal de comunicación entre un servidor y un cliente, el cual recibirá una serie de frames correspondientes a dos imágenes que le envía el servidor.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba implementada en Java para establecer la conexión.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

El puerto TCP del servidor, el puerto UDP y la dirección IP del cliente, y las imágenes a enviar.

- **Eventos a observar**

El cliente se comunica con el servidor, que le mandará una serie de frames correspondientes a las imágenes.

- **Consecuencias en caso de fallo**

No podremos transmitir las imágenes que capta la cámara a la aplicación cliente de nuestro sistema y, por lo tanto, éste no funcionará como debe.

Aplicación CMC (Coordenadas Mando Consola)

Detección del movimiento horizontal de los LED's

- **Descripción del evento**

Se va a comprobar que el driver creado para el Wiimote detecta movimientos horizontales de los LED's.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: driver implementado en Java para manejar el Wiimote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser la realización de un movimiento horizontal de uno o varios de los LEDs'.

- **Eventos a observar**

Se espera observar un cambio en los valores relativos al eje X que indican la posición de los LED's.

- **Consecuencias en caso de fallo**

Las consecuencias serían críticas, ya que en caso de fallo no seríamos capaces de detectar movimientos horizontales de los LED's y, por lo tanto, el sistema no funcionaría como debe.

Detección del movimiento vertical de los LED's

- **Descripción del evento**

Se va a comprobar que el driver creado para el Wiimote detecta movimientos verticales de los LED's.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: driver implementado en Java para manejar el Wiimote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser la realización de un movimiento vertical de uno o varios de los LED's.

- **Eventos a observar**

Se espera observar un cambio en los valores relativos al eje Y que indican la posición de los LED's.

- **Consecuencias en caso de fallo**

Las consecuencias serían críticas, ya que en caso de fallo no seríamos capaces de detectar movimientos verticales de los LED's y, por lo tanto, el sistema no funcionaría como debe.

Detección del movimiento en profundidad de los LED's

- **Descripción del evento**

Se va a comprobar que el driver creado para el Wiimote detecta movimientos en profundidad de los LED's.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: driver implementado en Java para manejar el Wiimote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser la realización de un movimiento en profundidad de uno o varios de los LED's.

- **Eventos a observar**

Se espera observar un cambio en los valores relativos al eje Z que indican la posición de los LED's.

- **Consecuencias en caso de fallo**

Las consecuencias serían críticas, ya que en caso de fallo no seríamos capaces de detectar movimientos en profundidad de los LED's y, por lo tanto, el sistema no funcionaría como debe.

Detección de la posición de varios LED's simultáneamente

- **Descripción del evento**

Se va a comprobar que el driver creado para el Wiimote detecta la posición de varios LED's simultáneamente.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: driver implementado en Java para manejar el Wiimote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser la activación de varios LED's simultáneamente.

- **Eventos a observar**

Se espera observar los datos relativos a la posición de cada uno de los LED's activados.

- **Consecuencias en caso de fallo**

La consecuencia sería que la información sobre la posición del usuario no sería tan exacta como la deseada y, por tanto, la simulación de la ventana virtual puede dar la sensación de ser falsa al no ajustarse bien con los movimientos del usuario.

Detección de la posición de los LED's en un entorno con otras fuentes de luz

- **Descripción del evento**

Se va a comprobar que el driver creado para el Wiimote detecta la posición de los LED's, incluso en entornos con fuentes de luz considerables externas al sistema.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: driver implementado en Java para manejar el Wiimote.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser la activación de los LED's.

- **Eventos a observar**

Se espera poder observar los datos relativos a la posición de cada uno de los LED's activados en este entorno hostil.

- **Consecuencias en caso de fallo**

En caso de que falle esta prueba, la consecuencia más inmediata sería que el sistema no puede usarse en entorno con fuentes de luz externas de una magnitud considerable, y, por tanto, debe de limitarse la existencias de estas fuentes externas o intentar usar LED's más potentes de forma que se puedan distinguir mejor.

LED's

Detección de máxima distancia de los LED's

- **Descripción del evento**

Se medirá la intensidad de los LED's a varias distancias para comprobar si son recibidos correctamente por el sensor.

Hardware: Wiimote, LED's y equipo de pruebas.

Software: WiiUseJ y Bluesoleil.

Personal: un ingeniero de pruebas.

- **Datos de entrada**

Los datos serán los propios LED's que apuntarán al sensor.

- **Eventos a observar**

Se espera observar hasta qué momento el sensor es capaz de detectar los LED's con suficiente precisión.

- **Consecuencias en caso de fallo**

El sensor no detectará los LED's.

Detección del máximo ángulo de incidencia

- **Descripción del evento**

Se medirá el máximo ángulo de incidencia de los LED's con respecto al sensor a partir del cuál éste no podrá detectarlos.

Hardware: Wiimote, LED's y equipo de pruebas.

Software: WiiUseJ y Bluesoleil.

Personal: un ingeniero de pruebas.

- **Datos de entrada**

Los datos serán los propios LED's que apuntarán al sensor.

- **Eventos a observar**

Se espera observar hasta qué momento el sensor es capaz de detectar los LED's con suficiente precisión.

- **Consecuencias en caso de fallo**

El sensor no detectará los LED's.

Algoritmo de recorte

Recorte de una imagen desde la esquina superior izquierda

- **Descripción del evento**

Tomaremos una imagen y la recortaremos desde la esquina superior izquierda, sin llegar a tocar los otros dos bordes.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba para recortar imágenes.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una imagen.

- **Eventos a observar**

Se espera obtener un recorte adecuado de la imagen.

- **Consecuencias en caso de fallo**

No se realizará bien el recorte o incluso no se podrá realizar.

Recorte de una imagen desde la esquina superior derecha

- **Descripción del evento**

Tomaremos una imagen y la recortaremos desde la esquina superior derecha, sin llegar a tocar los otros dos bordes.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba para recortar imágenes.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una imagen.

- **Eventos a observar**

Se espera obtener un recorte adecuado de la imagen.

- **Consecuencias en caso de fallo**

No se realizará bien el recorte o incluso no se podrá realizar.

Recorte de una imagen desde la esquina inferior izquierda

- **Descripción del evento**

Tomaremos una imagen y la recortaremos desde la esquina inferior izquierda, sin llegar a tocar los otros dos bordes.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba para recortar imágenes.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una imagen.

- **Eventos a observar**

Se espera obtener un recorte adecuado de la imagen.

- **Consecuencias en caso de fallo**

No se realizará bien el recorte o incluso no se podrá realizar.

Recorte de una imagen desde la esquina inferior derecha

- **Descripción del evento**

Tomaremos una imagen y la recortaremos desde la esquina inferior derecha, sin llegar a tocar los otros dos bordes.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba para recortar imágenes.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una imagen.

- **Eventos a observar**

Se espera obtener un recorte adecuado de la imagen

- **Consecuencias en caso de fallo**

No se realizará bien el recorte o incluso no se podrá realizar.

Recorte de una imagen en una zona central

- **Descripción del evento**

Tomaremos una imagen y la recortaremos en una zona central, sin tocar ningún borde.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba para recortar imágenes.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una imagen.

- **Eventos a observar**

Se espera obtener un recorte adecuado de la imagen.

- **Consecuencias en caso de fallo**

No se realizará bien el recorte o incluso no se podrá realizar.

3ª iteración

Integración: enviar órdenes

Generación de órdenes a partir de los movimientos del usuario

- **Descripción del evento**

La prueba consistirá en traducir los movimientos del usuario en órdenes para ser enviadas al servidor, el cual se encargará de mover la cámara remota.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: clase de prueba implementada en Java.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los movimientos horizontales, verticales y en profundidad de los LED's.

- **Eventos a observar**

La clase de prueba deberá generar los valores correspondientes a la posición del usuario a partir de las señales emitidas por los LED's infrarrojos.

- **Consecuencias en caso de fallo**

No se podrá determinar la posición del usuario en cada momento y, por lo tanto, no se podrá enviar órdenes al servidor.

Generación de órdenes a partir de movimientos simulados

- **Descripción del evento**

La prueba consiste en generar órdenes a partir de movimientos simulados, en los que se variará la supuesta posición del usuario.

Hardware necesario: equipo de pruebas.

Software necesario: clase de prueba implementada en Java.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Una serie de posiciones del usuario generados automáticamente.

- **Eventos a observar**

La clase de prueba traducirá dichas posiciones en órdenes que se mostrarán por la consola.

- **Consecuencias en caso de fallo**

La posición del usuario no se podrá traducir a órdenes, aunque, realmente, cuando nos debe funcionar es con movimientos reales del usuario, y no automáticos como los que usamos en esta prueba.

Envío de órdenes al servidor

- **Descripción del evento**

La prueba consiste en generar órdenes a partir de los movimientos del usuario y enviarlos al servidor para que éste se encargue de transferirlos a la cámara remota.

Hardware necesario: Wiimote, LED's y equipo de pruebas.

Software necesario: clase de prueba implementada en Java.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los movimientos horizontales, verticales y en profundidad de los LED's y los puertos y dirección IP del servidor.

- **Eventos a observar**

La clase de prueba traducirá cada posición del usuario en órdenes que serán enviadas al servidor.

- **Consecuencias en caso de fallo**

La posición del usuario no se podrá traducir a órdenes ni el servidor podrá recibir éstas para poder controlar la cámara remota.

4ª Iteración

Integración: envío órdenes / recepción frames

Captura de imágenes (frames) de forma local

- **Descripción del evento**

La prueba consistirá en capturar imágenes de forma local a través de la cámara web e irlas mostrando en una ventana.

Hardware necesario: cámara web y equipo de pruebas.

Software necesario: clase de prueba implementada en Java y librería JMF.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las imágenes que capturará la cámara web.

- **Eventos a observar**

La ventana de prueba debe mostrar las imágenes capturadas por la cámara web en todo momento.

- **Consecuencias en caso de fallo**

No se podrán visualizar, de ninguna forma, las imágenes que se deben mostrar al usuario del sistema y, por tanto, tampoco se podrá enviar frames al cliente.

Captura / recepción de imágenes (frames) de forma remota

- **Descripción del evento**

La prueba consistirá en capturar imágenes de forma local a través de la cámara web, transmitirlas a través de la red y mostrarlas en un equipo remoto.

Hardware necesario: cámara web y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java y librería JMF.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las imágenes que capturará la cámara web.

- **Eventos a observar**

La ventana de prueba del equipo remoto debe mostrar las imágenes capturadas por la cámara web en todo momento.

- **Consecuencias en caso de fallo**

Las imágenes no podrán ser visualizadas de forma remota, por lo que el sistema se debería restringir a un entorno local.

Envío de órdenes simuladas / captura de imágenes de forma remota

- **Descripción del evento**

La prueba consistirá en mostrar las imágenes capturadas de forma remota obtenidas a partir del envío de órdenes de manera simulada.

Hardware necesario: cámara web, base móvil TrackerPod y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF y driver de la base móvil TrackerPod.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las órdenes simuladas que se van a enviar.

- **Eventos a observar**

La ventana de prueba debe mostrar las imágenes capturadas por la cámara web en todo momento y de acuerdo a las órdenes enviadas.

- **Consecuencias en caso de fallo**

En caso de fallo, no será posible visualizar las imágenes capturadas por la cámara de manera dinámica (mediante el envío de órdenes de movimiento a la base PTZ).

Envío de órdenes reales / captura de imágenes de forma remota

Descripción del evento

La prueba consistirá en mostrar las imágenes capturadas de forma remota obtenidas a partir del envío de órdenes reales como consecuencia del movimiento del usuario del sistema.

Hardware necesario: Wiimote, cámara web, base móvil TrackerPod, gafas con LED's y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF, librería WiiUseJ y driver de la base móvil TrackerPod.

Personal necesario: un ingeniero de pruebas.

Datos de entrada

Los datos de entrada van a ser las órdenes reales derivadas del movimiento del usuario.

Eventos a observar

La ventana de prueba debe mostrar las imágenes capturadas por la cámara web en todo momento y de acuerdo al movimiento del usuario.

Consecuencias en caso de fallo

No será posible visualizar las imágenes capturadas por la cámara de acuerdo al movimiento del usuario.

Integración y calibración del sistema

Detección cámara-ordenador

Descripción del evento

Esta prueba consistirá en comprobar que la detección, a través de la librería JMF, de la cámara web Logitech QuickCam Pro 9000 se lleva a cabo de forma correcta en el ordenador que actuará como servidor.

Hardware necesario: cámara web Logitech QuickCam Pro 9000 y equipo servidor.

Software necesario: clase de prueba implementada en Java y librería JMF.

Personal necesario: un ingeniero de pruebas.

Datos de entrada

Los datos de entrada van a ser las imágenes que capturará la cámara web Logitech QuickCam Pro 9000.

Eventos a observar

La ventana de prueba debe mostrar las imágenes capturadas por la cámara web Logitech QuickCam Pro 9000 en todo momento.

Consecuencias en caso de fallo

No se podrá usar la cámara web Logitech QuickCam Pro 9000 en el sistema real, debido a que JMF no es capaz de detectarla y no será capaz de devolver las imágenes capturadas por ésta.

Integración de órdenes reales / captura de imágenes / informe de log

Descripción del evento

La prueba consistirá en probar que la integración del informe de log en el sistema se ha llevado a cabo de forma correcta y sin afectar al funcionamiento de la funcionalidad existente.

Hardware necesario: cámara web Logitech QuickCam Pro 9000, base móvil TrackerPod, mando Wiimote, gafas con LED's y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF, librería WiiUseJ y driver de la base móvil TrackerPod.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las órdenes reales derivadas del movimiento del usuario junto con la simulación de eventos de error en el sistema.

- **Eventos a observar**

La ventana de prueba debe mostrar, además de las imágenes capturadas por la cámara web, la información sobre errores que se produzcan en el sistema durante su funcionamiento.

- **Consecuencias en caso de fallo**

En caso de fallo, no será posible informar al usuario de errores que se produzcan en el sistema.

Integración de algoritmos de recorte y zoom

- **Descripción del evento**

La prueba consistirá en comprobar que los algoritmos de recorte y zoom implementados realizan su función de forma correcta y su integración en el sistema no afecta a la funcionalidad anterior existente.

Hardware necesario: cámara web Logitech QuickCam Pro 9000, base móvil TrackerPod, mando Wiimote, gafas con LED's y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF, librería WiiUseJ y driver de la base móvil TrackerPod.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las órdenes reales derivadas del movimiento del usuario.

- **Eventos a observar**

La ventana de prueba debe mostrar las imágenes procesadas que le llegan del servidor. Este procesamiento debe corresponder al uso de los algoritmos de recorte y zoom sobre las imágenes originales capturadas por la cámara web.

- **Consecuencias en caso de fallo**

En caso de fallo, no se podrán procesar las imágenes de forma correcta con el objetivo de simular tanto el desplazamiento lateral como el zoom de las imágenes capturadas.

Funciones avanzadas de la ventana de usuario

- **Descripción del evento**

Esta prueba se basa en probar las funciones avanzadas disponibles en la ventana de usuario: configuración especifica de los puertos a usar y dirección IP del servidor, captura por pantalla de la ventana virtual …

Hardware necesario: cámara web Logitech QuickCam Pro 9000, base móvil TrackerPod, mando Wiimote, gafas con LED's y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF, librería WiiUseJ y driver de la base móvil TrackerPod.

Personal necesario: un ingeniero de pruebas.

- **Datos de entrada**

Los datos de entrada van a ser las acciones solicitadas por el usuario en la ventana de la aplicación cliente.

- **Eventos a observar**

Se espera observar que el sistema responde correctamente a las acciones solicitadas.

- **Consecuencias en caso de fallo**

Si la prueba falla en alguna de las acciones solicitadas, tendrá como consecuencias que el usuario no podrá disponer de dicha funcionalidad en el sistema.

Calibración del sistema completo

Descripción del evento

La prueba consistirá en comprobar que la calibración del sistema al completo sea correcta; para ello, nos basaremos en el concepto de prueba y error, es decir, se irán modificando los parámetros de la calibración hasta conseguir simular lo que vería el usuario de una forma más o menos precisa. En esta prueba, estarán todos los integrantes de grupo para tener una visión lo más objetiva posible de la calibración.

Hardware necesario: cámara web Logitech QuickCam Pro 9000, base móvil TrackerPod, mando Wiimote, gafas con LED's y dos equipos de pruebas.

Software necesario: clases de prueba implementadas en Java, librería JMF, librería WiiUseJ y driver de la base móvil TrackerPod.

Personal necesario: todos los integrantes del grupo.

Datos de entrada

Los datos de entrada van a ser las órdenes reales derivadas del movimiento del usuario.

Eventos a observar

La ventana de prueba debe mostrar las imágenes, una vez procesadas mediante los algoritmos de recorte y zoom, que llegan del servidor como consecuencia de los movimientos del usuario.

Consecuencias en caso de fallo

En caso de fallo (no conseguir una buena calibración), las consecuencias serán que las imágenes que veremos en la ventana no se corresponderán a los movimientos del usuario.