Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema móvil de reconocimiento de objetos y posicionamiento en una exposición basado en realidad aumentada

por
Carlos Arturo Sánchez Uribe
Matrícula 207201489

Asesor: M. En C. Oscar Alvarado Nava
Profesor Asociado
Departamento de Electrónica

Trimestre 11-O

Índice

1. Introducción

En la última década los dispositivos móviles dejaron de ser simples objetos, que solo se utilizaban para realizar llamadas telefónicas o reproducir música, para convertirse en computadoras de bolsillo con múltiples funcionalidades, desde hacer video-llamadas o ejecutar aplicaciones de red especializadas. Esto significa un mayor reto para crear aplicaciones donde se aplique inteligencia artificial, en particular visón de computadora . En años recientes el desarrolló de la tecnología ha traído muchos avances como lo es la Realidad Aumentada(RA)[1]. Se ha investigado acerca de como la RA podría realzar la experiencia del usuario en aplicaciones existentes o aplicaciones específicamente creadas para este propósito.

Un sistema basado en realidad aumentada genera una vista compuesta para el usuario, la cual es una combinación de la escena real vista por el usuario y una escena virtual generada por la computadora que aumenta la escena con información adicional. El sistema que se desarrollo, permite al usuario de un dispositivo iOS[1] hacer una conexión a un servidor dentro del museo, cuando el usuario lo desee puede tomar una foto de algún objeto en exhibición y este mandará la imagen al servidor,el cual, analizará la foto y si detecta algún objeto conocido entonces lo señalara, dibujando un perímetro alrededor de este y lo enviará de regreso al cliente móvil junto con la información del objeto analizado, ya recibida la imagen se despliega en la pantalla del dispositivo y es cuando está listo para tomar alguna foto más.

[1]Sistema operativo exclusivo de los dispositivos: iPhone, iPad y iPodTouch.

Este sistema enriquece la experiencia del usuario en su visita al museo, permitiendo conocer más a fondo las piezas exhibidas, obras de arte, pinturas etc., de una forma más interactiva y novedosa.

2. Antecedentes

En 1968 Ivan Sutherland crea el primer sistema de realidad aumentada, que es también el primer sistema de realidad virtual . Utiliza una pantalla transparente colocada en la cabeza que es seguida por uno o dos seguidores 6DOF (el cual se compone de un seguidor mecánico y un seguidor ultrasónico). Debido al bajo poder de procesamiento de las computadoras en la época, solo muy pocas lineas de dibujo podían ser mostradas en tiempo real. En 1992 Tom Caudell y David Mizell acuñan el termino de Realidad Aumentada. En el mismo año IBM introduce el primer *smartphone*[2] el *Simon Personal Communicator* el cual tenía 1 Megabyte de memoria y una pantalla sensible al tacto en blanco y negro con una resolución de 160 x 293 pixeles, pesaba 500 gramos y servía como teléfono, calculadora, libreta de direcciones, fax y podía enviar y recibir correo electrónico.

En 1997 Steve Feiner presenta la *Touring Machine*, el cual es el primer sistema de realidad aumentada móvil. Utiliza una pantalla transparente que se monta en la cabeza, una mochila la cual contiene una computadora, un GPS diferencial y una computadora de mano con una interfaz táctil.

En el 2001 Jürgen Fruend presenta el *AR-PDA*, el cual es un concepto de como se debe implementar un sistema de RA inalámbrico y un prototipo especial tipo palm que cabe en las manos. En 2004 Mathias Möhring presenta un sistema que rastrea marcadores 3D con un teléfono celular así se pudo demostrar el primer sistema de RA en un celular de consumo. En 2008 METAIO presenta una guía de RA para la muestra de arte Islámico la cual utiliza un sistema de seguimiento natural.

[2]Término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono móvil común.

3. Descripción Tecnica

El sistema total se dividió en módulos los cuales fueron desarrollados en 4 etapas.

3.1. Etapa 1. Los módulos desarrollados son

- La creación del cliente móvil que tenga acceso a la cámara del dispositivo iOS.
- Desarrollo de los métodos que permitan la captura de una imagen y la funcionalidad de recibir y enviar una imagen desde y al servidor.

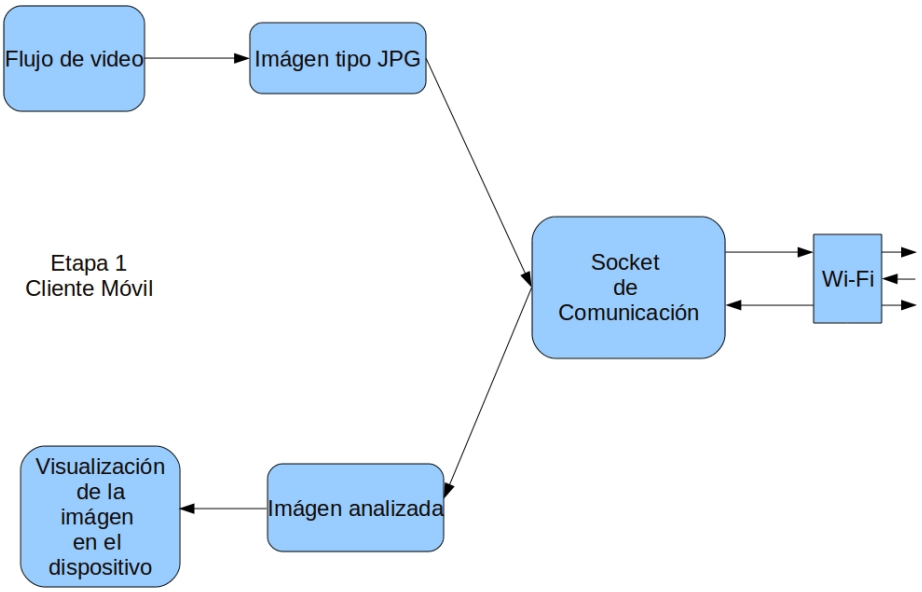

Figura 1. Muestra la Etapa 1.

3.2. Etapa 2. Los módulos desarrollados son:

- La creación del servidor que recibirá y analizara las imágenes tomadas.

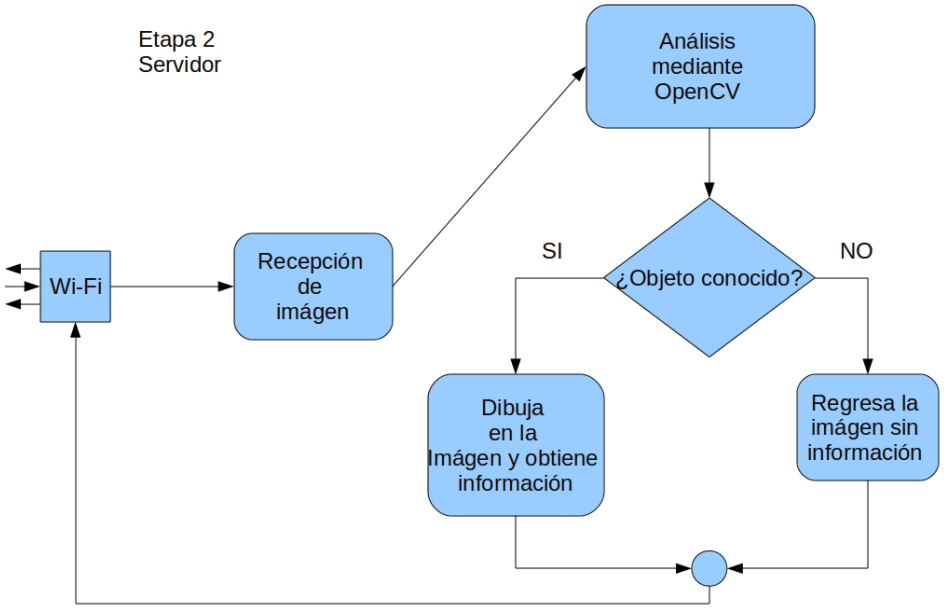

Figura 2. Muestra la Etapa 2.

3.3. Etapa 3. Los módulos desarrollados son:

- Entrenamiento de la red-neuronal que analizará las imágenes y detectará los objetos de la exhibición.

3.4. Etapa 4. Los módulos desarrollados son:

- Integración del sistema.
- Realizar pruebas y detectar posibles fallos.

4. Desarrollo

En este apartado se explicara el código fuente generado durante las etapas 1 y 2.

4.1. Cliente móvil. El cliente móvil fue desarrollado enteramente en Objective-C[3] el cual es un lenguaje basado en el modelo vista controlador, lo cual implica que siempre haya un controlador *.h y su vista *.m. Su desarrollo implicó la utilización de las librerías smallsockets[4][2] gracias a esto la implementación del modulo de comunicación fue de una forma más sencilla. La captura de video se hizo mediante el método AVCapture el cual permite la obtención directa de video desde la cámara del dispositivo y su codificación directa a JPEG u otros formatos nativos del iOS. La figura 3 detalla los módulos a desarrollar.

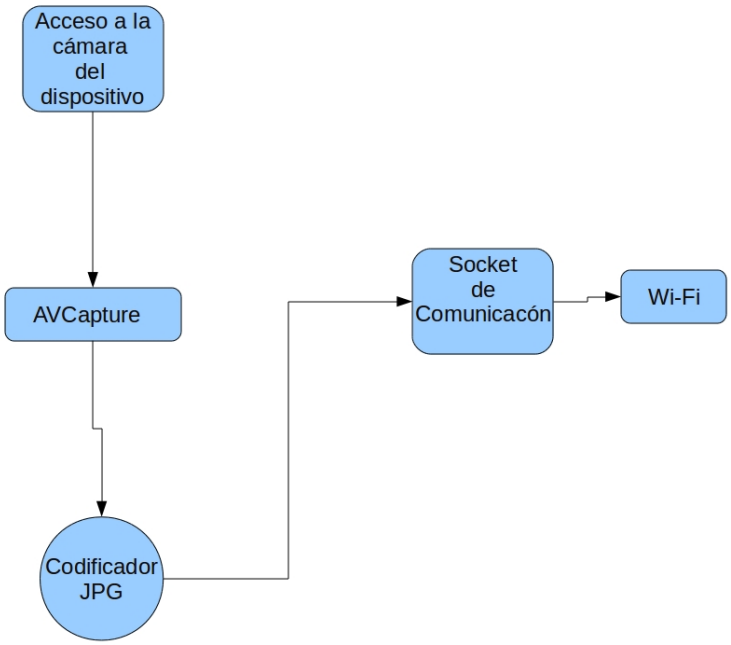

Figura 3. Muestra el desarrollo del cliente móvil.

Listing 1. PTVideoViewController.h

```
@interface PT_VideoViewController : UIViewController<NSStreamDelegate>
        UIView *vistaSobrepuesta;
```

[3]Lenguaje de programación orientado a objetos creado como un superconjunto de C pero que implementa un modelo de objetos basado en Smalltalk.

[4]Envoltura para Mac OSX en Objective-C de sockets tipo BSD

```
UILabel *Informacion;
 AVCaptureStillImageOutput *ImagenSalida;
UIImageView *vImage;
}
```

Declaración de variables:

- vistaSobrepuesta: Vista que permite el despliegue de video desde el dispositivo.
- Informacion: Etiqueta donde se mostrará la información obtenida del objeto detectado.
- ImagenSalida: Variable que permite el despliegue de la imagen analizada.
- vImage: Vista donde se desplegara la imagen.

Listing 2. PTVideoViewController.h

```
@property (nonatomic, retain) IBOutlet UIView *vistaSobrepuesta;
@property (nonatomic, retain) IBOutlet UILabel *Informacion;
@property (nonatomic, retain) AVCaptureStillImageOutput *ImagenSalida;
@property (nonatomic, retain) IBOutlet UIImageView *vImage;
```

Declaración de propiedades, todas las variables, botones, vistas, etc., Llevan su *property*.

Listing 3. PTVideoViewController.h

```
-(IBAction)Captura;
-(void) iniComuSmall:(NSData*)imagen;
-(void) iniServidor;
-(void) iniServidorInformacion;
@end
```

Declaración de métodos:

- Captura: Método que captura una imagen del flujo de video, mediante la acción de un botón.
- iniComuSmall: Método que envía la imagen capturada al servidor para analizarla, como parámetro recibe la imagen capturada del flujo de video.
- iniServidor: Método que recibe la imagen analizada y la muestra en la pantalla del dispositivo.
- iniServidorInformacion: Método que recibirá la información del objeto.
- @end: termina el archivo PTVideoViewController.h

Listing 4. PTVideoViewController.m

```
@implementation PT_VideoViewController
@synthesize vistaSobrepuesta;
@synthesize Informacion;
@synthesize ImagenSalida;
@synthesize vImage;

- (void)dealloc {
    [PT_VideoViewController release];
    [ImagenSalida release];
    [vImage release];
    [vistaSobrepuesta release];
    [Informacion release];
    [super dealloc];
}
```

Todas las variables declaradas en el *.h tienen que ser sintetizadas y también tienen que ser desechadas al final del programa con el método dealloc. *Implementation* hace referencia a la clase *.h.

Listing 5. PTVideoViewController.m

```
UIButton *boton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[boton addTarget:self action:@selector(Captura)
forControlEvents:UIControlEventTouchDown];
[boton setTitle:@"Captura" forState:UIControlStateNormal];
boton.frame = CGRectMake(100, 170, 100, 30);
[self.view addSubview:boton];
```

En el método viewDidAppear:

- Creación del botón de captura el cual ejecutara el método Çaptura".

Listing 6. PTVideoViewController.m

```
AVCaptureSession *sesion = [[AVCaptureSession alloc] init];
sesion.sessionPreset =
AVCaptureSessionPreset640x480;
CALayer *CapaVista = self.vistaSobrepuesta.layer;
NSLog(@"viewLayer = %@",CapaVista);

AVCaptureVideoPreviewLayer *VistaPreviaVideo =
[[AVCaptureVideoPreviewLayer alloc] initWithSession:sesion];
```

```
VistaPreviaVideo.frame = self.vistaSobrepuesta.bounds;

[self.vistaSobrepuesta.layer addSublayer:VistaPreviaVideo];

AVCaptureDevice *dispositivo =
[AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

NSError *error = nil;
AVCaptureDeviceInput *entrada =
[AVCaptureDeviceInput deviceInputWithDevice:dispositivo error:&error];
if(!entrada) {
        NSLog(@"Error Tratando de abrir la camara: %@",error);
        }

[sesion addInput:entrada];
ImagenSalida = [[AVCaptureStillImageOutput alloc] init];
NSDictionary *Salidas =
[[NSDictionary alloc] initWithObjectsAndKeys: AVVideoCodecJPEG,
AVVideoCodecKey, nil];

[ImagenSalida setOutputSettings:Salidas];
[sesion addOutput:ImagenSalida];
[sesion startRunning];
```

En el método viewDidAppear: Se programa la captura de video, con formato JPEG, la cual se colocara en la vistaSobrepuesta. Si el dispositivo no cuenta con cámara, como es el caso del iPad 1, iPod Touch de 1ra-3ra generación, entonces manda un error y no se inicia la aplicación. Al final se inicia la sesión de video.

Listing 7. PTVideoViewController.m

```
- (void) iniComuSmall:(NSData *)imagen{
    Socket *socket;
    int port = 3499;
    NSString *host = @"172.20.10.4";

    socket = [Socket socket];

    @try {
        [socket connectToHostName:host port:port];
```

```
        [socket writeData:imagen];
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
    }

    [socket close];
}
```

Método iniComuSmall: Este método envía la imagen obtenida del flujo de video.

- Se crea el socket de comunicación y se configura.
- Se hace la conexión con los datos configurados.
- Si existe algún error en la conexión se manda un error.
- Al final se cierra la conexión.

Listing 8. PTVideoViewController.m

```
- (void)iniServidor{

    Socket *socket;
    NSMutableData *respuesta = [[[NSMutableData alloc]
    init] autorelease];
    NSString *resultado;

    socket =[[Socket alloc] init];
    [socket listenOnPort:3498];

    @try {
        [socket acceptConnection];
        while ([socket readData:respuesta]) {
        NSLog(@"Recibiendo Imagen");
        }
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
    }
    [socket close];
```

```
    UIImage *imagen = [[UIImage alloc] initWithData:respuesta];
    self.vImage.image = imagen;

}
```

Método iniServidor: Método que inicia el Servidor para recibir la imagen enviada desde el cliente remoto.

- Se crea el socket de comunicación.
- Se crea una variable donde se van a guardar los datos que se envían desde el servidor.
- Se acepta la conexión y mientras se reciba información el socket permanece abierto.
- Si existe algún error en la conexión se manda un error.
- Se cierra la conexión.
- Al final los datos obtenidos se convierten en imagen y se muestran al usuario.

Listing 9. PTVideoViewController.m

```
- (void)iniServidorInformacion{

    Socket *socket;
    NSMutableData *respuesta = [[[NSMutableData alloc]
    init] autorelease];
    NSString *resultado;
    socket =[[Socket alloc] init];
    [socket listenOnPort:3497];

    @try {
        [socket acceptConnection];
        while ([socket  readData:respuesta])
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
    }
    [socket close];
    resultado = [[NSString alloc] initWithData:respuesta
    encoding:NSASCIIStringEncoding];
    Informacion.text = resultado;
}
```

Método iniServidorInformacion: Método que inicia el Servidor para recibir la información del objeto detectado

- Se crea el socket de comunicación.
- Se crea una variable donde se van a guardar los datos que se envían desde el servidor.
- Se acepta la conexión y mientras se reciba información el socket permanece abierto.
- Si existe algún error en la conexión se manda un error.
- Se cierra la conexión.
- Al final los datos obtenidos se convierten a un tipo NSString y se muestran en la etiqueta Información

.

Listing 10. PTVideoViewController.m

```
-(IBAction) Captura{
AVCaptureConnection *ConexVideo = nil;
      for (AVCaptureConnection *conexion in ImagenSalida.connections){
         for (AVCaptureInputPort *puerto in [conexion inputPorts]){
                  if ([[puerto mediaType] isEqual:AVMediaTypeVideo] ){
                                    ConexVideo = conexion;
                                    break;
                  }
         }
        if (ConexVideo) { break; }
         }

NSLog(@"Pedimento de captura de video: %@", ImagenSalida);
[ImagenSalida captureStillImageAsynchronouslyFromConnection:
ConexVideo completionHandler: ^(CMSampleBufferRef imageSampleBuffer,
NSError *error)
 {

    CFDictionaryRef exifAttachments =
    CMGetAttachment( imageSampleBuffer, kCGImagePropertyExifDictionary,
    NULL);
          if (exifAttachments){
                  NSLog(@"Datos: %@", exifAttachments);
          }
          else
                  NSLog(@"Sin Datos");
```

```
                NSData *DatosImagen =
                [AVCaptureStillImageOutput
                 jpegStillImageNSDataRepresentation:
                imageSampleBuffer];


        [self iniComuSmall:DatosImagen];
        [self iniServidor];
        [self iniServidorInformacion];
        }];
}
```

Método Captura: Metodo que es accionado por el botón de captura, el cual se encarga de tomar una foto del stream de video y la manda al servidor el cual analiza la foto y la devuelve.

- Se hace una captura directamente del flujo de video
- La variable exifAttachments obtiene datos de la imagen capturada y los muestra en consola
- La imagen capturada se convierte en datos con codificación jpeg y así se puede mandar mediante el socket.
- Se manda llamar al metodo iniComuSmall, como parametro se envía la imagen capturada.
- Se inicializa el servicio que recibirá la imagen.
- Se inicializa el servicio que recibirá la información de la imagen.

4.2. Servidor. El servidor fue desarrollado en Java el cual espera en el fondo a que el dispositivo le envíe una imagen. Ya que recibe la imagen la guarda en la ruta */home/usuario/PT/Imagenes* y ejecuta el programa que la analizara. El servidor se desarrollo en java para probar la interoperabilidad del proyecto. Para evitar que el usuario tuviera que ejecutar los programas manualmente, puesto que el servidor esta desarrollado en Java, el analizador en C++ y el cliente de salida también en Java, se implementaron los metodos: Runtime.getRuntime().exec() en Java y System() en C++. Estos métodos ejecutan los comandos que se les manden como parámetros lo cual hace que desde el programa principal se ejecuten todos los demás sin ningún problema. La figura 4 muestra las partes que debe llevar el Servidor.

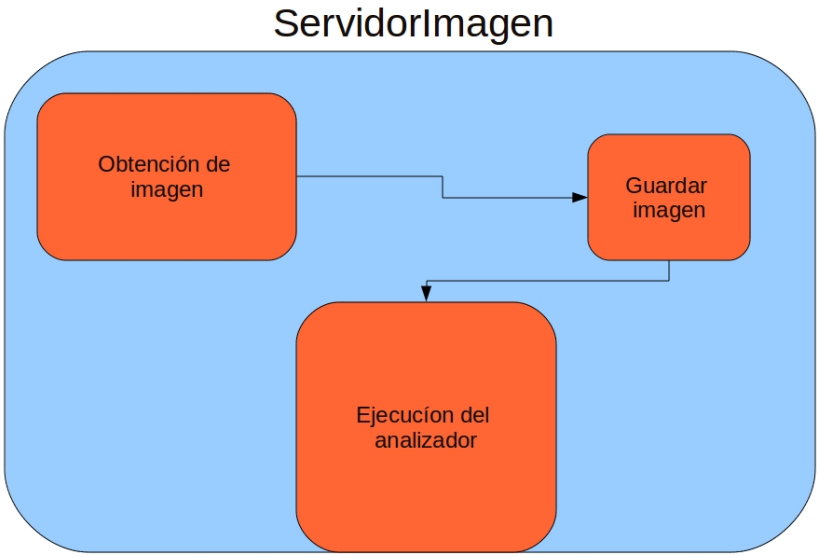

Figura 4. Muestra el desarrollo del servidor.

Listing 11. ServidorImagen.java

```
ServerSocket servsock = new ServerSocket(3499);
    long inicio = System.currentTimeMillis();
    int BytesLeidos;
    int actual = 0;
    int tamarch=6022386;
```

Creación del socket.

Listing 12. ServidorImagen.java

```
String usuario = System.getProperty("user.name");
```

Linea que obtiene el nombre de usuario del sistema.

Listing 13. ServidorImagen.java

```
while (true) {
      System.out.println("\nEsperando...");

      Socket sock = servsock.accept();
      System.out.println("Coneccion Aceptada : " + sock);

    byte [] ArregloBytes = new byte [tamarch];
    InputStream is = sock.getInputStream();
    FileOutputStream fos =
    new FileOutputStream("/home/"+usuario+"/PT/Imagenes/objeto.jpg");
    BufferedOutputStream bos = new BufferedOutputStream(fos);
    BytesLeidos = is.read(ArregloBytes,0,ArregloBytes.length);
    actual = BytesLeidos;

    do {
       BytesLeidos = is.read(ArregloBytes, actual,
                             (ArregloBytes.length-actual));
       if(BytesLeidos >= 0) actual += BytesLeidos;
    } while(BytesLeidos > -1);

    bos.write(ArregloBytes, 0 , actual);
    bos.flush();
    long fin = System.currentTimeMillis();
    System.out.println(fin-inicio);
    bos.close();
    sock.close();

       File wd = new File("/home/"+usuario+"/PT");
       System.out.println(wd + "\n");
       Process proc = null;
       String comando[] = {"gnome-terminal","-e","./Proyecto"};
```

```
try {
    proc = Runtime.getRuntime().exec(comando, null, wd);
    System.out.println(proc);
}
catch (IOException e) {
    e.printStackTrace();
}

if (proc != null) {
    System.out.print(proc.getErrorStream());

try {
    proc.waitFor();
    proc.destroy();
}
catch (Exception e) {
    e.printStackTrace();
}
}
}
```

Método que espera alguna conexión, guarda la imagen y ejecuta el programa que analizará la imagen.

- El socket está en espera hasta que algún cliente remoto envía una imagen.
- Se crea el archivo tipo imagen.
- Se recibe el archivo.
- Se abre la terminal y se ejecuta el programa de reconocimiento con la imagen obtenida.

4.3. Analizador. El programa Proyecto.cpp es ejecutado desde el programa ServidorImagen.java, el programa abre la imagen y la analiza con la primera cascada, si no reconoce nada, intenta con la segunda cascada, si tampoco reconoce nada intenta con la tercer cascada sino encuentra nada ejecuta el programa ClienteImagen.java para que este mande la imagen de regreso al dispositivo móvil y envíe un mensaje de que no encontró nada en esa imagen. De igual forma si encontró algún objeto entonces dibuja sobre la imagen un perímetro y ejecuta el programa ClienteImagen.java para que este mande la imagen de regreso al dispositivo móvil y envíe información acerca del objeto. La figura 5 muestra las partes internas del analizador.

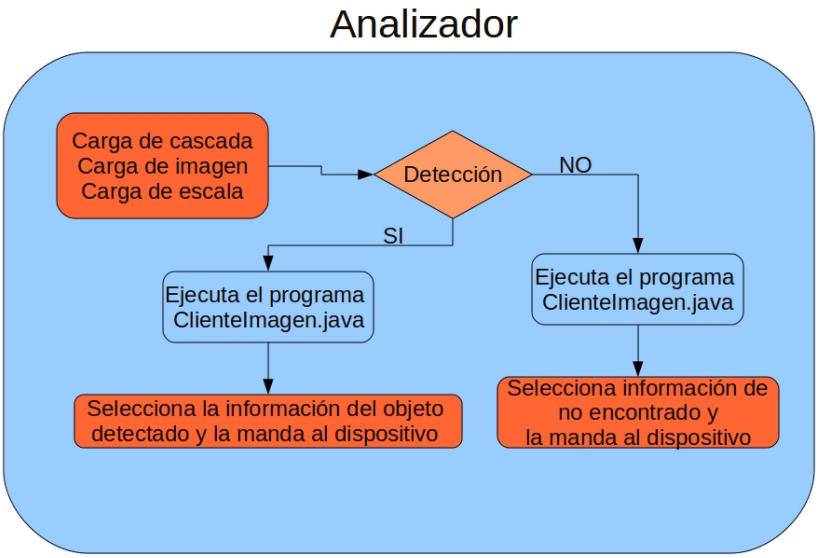

Figura 5. Muestra el desarrollo del analizador.

Listing 14. Proyecto.cpp

```
resultado = Objeto1();
    if(!resultado == 0){
        system("java ClienteImagen");
        i = 1;
       recibir(i);
       exit(0);
    }
```

- La función Objeto se manda llamar si regresa un 1 entonces detecto algún objeto lo enviará al dispositivo móvil y ejecutara la función que enviara la información respectiva al cliente móvil.
- Si el resultado es 0 se pasa a la siguiente función Objeto.

Listing 15. Proyecto.cpp

```
int Objeto1(){

    Mat Imagen;
    int resultado = 0, mensaje = 0;
    String usuario = getenv("USER");
    String cascadaCocacola =
    "/home/"+usuario+"/PT/Entrenamiento/cocacola.xml";
    CascadeClassifier cascada;
    double escala = 1;

    cascada.load(cascadaCocacola);
    if(cascada.empty())
        cout << "La cascada no se pudo cargar" << endl;

    cout << "La cascada que se utilizara es: "
    << cascadaCocacola << endl;
    escala = 1.5;
    cout << "La escala que se utilizara es: " << escala << endl;
    Imagen = imread("Imagenes/objeto.jpg", 1);
    cout << "La imagen que se analizara es objeto.jpg" << endl;

    resultado = DetectaryDibujar(Imagen, cascada, escala, mensaje);

    if(resultado == 0)
        return 0;
    else
        return 1;
}
```

- La función Objeto carga la cascada, obtiene el nombre de usuario, carga la escala y carga la imagen a analizar.
- Finalmente se manda llamar la función que detecta y dibuja, si esta función detecta algo en la foto, regresa un 1 sino regresará un 0.

Listing 16. Proyecto.cpp

```
int DetectaryDibujar(Mat& img, CascadeClassifier& cascad,
                     double scale, int msj)
{
    int i = 0, fallo = 0;
    double t = 0;
    vector<Rect> caras;
    const static Scalar colores[] = {CV_RGB(0,0,255),
CV_RGB(0,128,255), CV_RGB(0,255,255), CV_RGB(0,255,0),
 CV_RGB(255,128,0), CV_RGB(255,255,0), CV_RGB(255,0,0),
CV_RGB(255,0,255)};

    Mat gris, smallImg(cvRound (img.rows/scale),
cvRound(img.cols/scale), CV_8UC1);

    cvtColor(img, gris, CV_BGR2GRAY);
    resize(gris, smallImg, smallImg.size(), 0, 0, INTER_LINEAR);
    equalizeHist(smallImg, smallImg);

    t = (double)cvGetTickCount();

    cascad.detectMultiScale( smallImg, caras, 1.1, 2,
    0|CV_HAAR_SCALE_IMAGE, Size(30, 30) );
    if(caras.empty()){
        cout << "No se detecto nada" << endl;
        fallo = 0;
        if(msj == 1){
            cv::imwrite("Imagenes/resultado.jpg", img);
            system("java ClienteImagen");
            return fallo;
        }
        else{
            return fallo;
        }
        }
    else {
        fallo = 1;
    t = (double)cvGetTickCount() - t;
    printf("Tiempo = %g ms\n",
```

```
        t/((double)cvGetTickFrequency()*1000.));
    for(vector<Rect>::const_iterator r = caras.begin();
                                r != caras.end(); r++, i++)
    {
        Point centro;
        Scalar color = colores[i%8];
        int radio;
        centro.x = cvRound((r->x + r->width*0.5)*scale);
        centro.y = cvRound((r->y + r->height*0.5)*scale);
        radio = cvRound((r->width + r->height)*0.25*scale);
        circle(img, centro, radio, color, 3, 8, 0);
    }
    cv::imwrite("Imagenes/resultado.jpg", img);
    system("java ClienteImagen");
 }
    return fallo;
}
```

La función DetectaryDibujar.

- El método detectMultiscale de la libreria OpenCV[3] es tipo void por lo cual no regresa nada haya o no detectado algo en la imagen por lo tanto se utilizó el vector caras. Si el vector caras regresa vacío no se detectó nada, si el vector caras contiene algo(triángulos) entonces si se detectó algún objeto por lo tanto manda a dibujar en la imagen un círculo.
- El método imwrite de la librería OpenCV crea la imagen con el círculo dibujado y se manda llamar al programa ClienteImagen.java el cual enviará la imagen de regreso al dispositivo móvil.

4.4. Cliente Sock. El Sock.h es ejecutado desde el programa Proyecto.cpp (después de ejecutar el ClienteImagen), el cliente entonces tomará como parámetro el número del objeto detectado y seleccionará la información respectiva a ese objeto detectado y la enviará al dispositivo móvil. La figura 6 muestra la parte interna del programa sock.h.

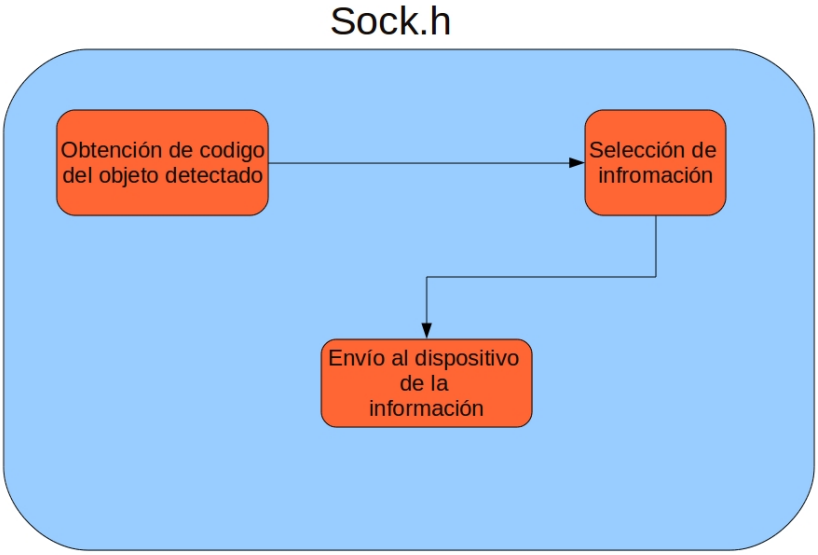

Figura 6.

Listing 17. Proyecto.cpp

```
void recibir(int i)
{
    int sockfd, puerto, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char cocacola[] = "es.wikipedia.org/wiki/Coca-Cola";
    char monalisa[] = "es.wikipedia.org/wiki/La_Gioconda";
    char cubo[] = "es.wikipedia.org/wiki/Cubo_de_Rubik";
    char buffer[] = "No hay datos disponibles";
    puerto = 3497;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
    if (sockfd < 0)
        error("Error al abrir el socket");
    server = gethostbyname("192.168.1.86");
    if (server == NULL) {
        fprintf(stderr,"Error host no valido\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
    (char *)&serv_addr.sin_addr.s_addr, server->h_length);
    serv_addr.sin_port = htons(puerto);
    if (connect(sockfd,(struct sockaddr *)
       &serv_addr,sizeof(serv_addr)) < 0)
        error("Error conectando");
    switch (i){
        case 1:
                n = write(sockfd,cocacola,strlen(cocacola));
                break;
        case 2:
                n = write(sockfd,monalisa,strlen(monalisa));
                break;
        case 3:
                n = write(sockfd,cubo,strlen(cubo));
                break;
        case 0:
                n = write(sockfd,buffer,strlen(buffer));
                break;
    }

    if (n < 0)
         error("Error al escribir en el socket");
    close(sockfd);
}
```

- Se creal el socket.
- Se obtiene el código del objeto detectado.
- Se envía la información del objeto.

4.5. ClienteImagen. El ClienteImagen.java es ejecutado desde el programa Proyecto.cpp, el cliente entonces tomará la imagen analizada y la enviará de regreso, al dispositivo móvil. Este programa se ejecuta inmediatamente después de la detección del objeto, se haya detectado algo o no. La figura 7 muestra la parte interna del programa ClienteImagen.java

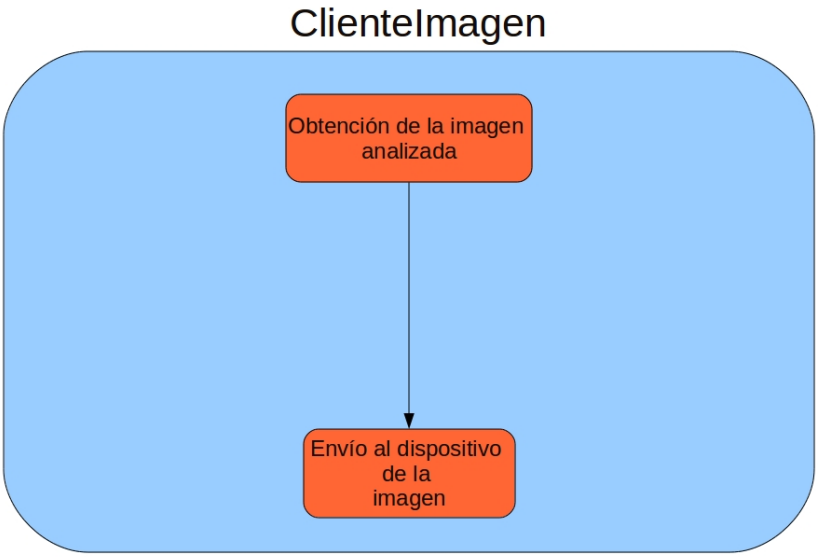

Figura 7.

Listing 18. ClienteImagen.java

```
public class ClienteImagen{
  public static void main (String [] args ) throws IOException {

    Socket sock = new Socket("192.168.1.86",3498);
    System.out.println("Conectando...");

      File Archivo = new File ("Imagenes/resultado.jpg");
      byte [] ArregloBytes  = new byte [(int)Archivo.length()];
      FileInputStream fis = new FileInputStream(Archivo);
      BufferedInputStream bis = new BufferedInputStream(fis);
      bis.read(ArregloBytes,0,ArregloBytes.length);
      OutputStream os = sock.getOutputStream();
      System.out.println("Enviando...");
      os.write(ArregloBytes,0,ArregloBytes.length);
      os.flush();
      sock.close();
      }

}
```

- Se creal el socket.
- Se conecta al dispositivo móvil, y comienza a enviar la imagen.
- Al terminar se cierra el socket.

5. Instalación

La instalación de todo el proyecto se hace mediante un *shell script*[5] el cual descarga: dependencias necesarias, herramientas para su utilización, OpenCV-2.2.0, compila los códigos fuente y ejecuta el programa ServidorImagen.java para comenzar a trabajar inmediatamente.

5.1. Script de instalación en Ubuntu. La instalación fue probada en Ubuntu[6] 11.04

Listing 19. Instalacion.sh

```
#!/bin/bash
##Instalacion de OpenCV 2.2 y Python.
##Compilacion e inicializacion del servidor
##que recibe las imagenes para ser analizadas.
##Compilacion del programa en C++ que analizara las imagenes.
##Compilacion del cliente que enviara las imagenes analizadas
##de regreso al dispositivo.

OURL="http://downloads.sourceforge.net/project/opencvlibrary/
opencv-unix/2.2/OpenCV-2.2.0.tar.bz2"
DLHOME=".."
ODEST="OpenCV-2.2.0/"
PROYECTO="PT"
#######Instalacion de dependencias
        sudo apt-get install build-essential libgtk2.0-dev
libjpeg62-dev libtiff4-dev libjasper-dev libopenexr-dev
cmake python-dev python-numpy libtbb-dev libeigen2-dev
yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev
libtheora-dev libvorbis-dev libxvidcore-dev
#######Obtencion de OpenCV 2.2
        cd "$DLHOME"
        wget $OURL
        tar -xvf OpenCV-2.2.0.tar.bz2
        cd $ODEST
        cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON
-D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON
-D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON .
```

[5]Es una interfaz de texto de altas prestaciones, que sirve fundamentalmente para tres cosas: administrar el sistema operativo, lanzar aplicaciones (e interactuar con ellas) y como entorno de programación.

[6]Ubuntu es un sistema operativo mantenido por Canonical y la comunidad de desarrolladores. Utiliza un núcleo Linux, y su origen está basado en Debian.

```
 && make && sudo make install
########Configuracion de librerias
        sudo chmod 766 /etc/ld.so.conf.d/opencv.conf
         echo '/usr/local/lib' > /etc/ld.so.conf.d/opencv.conf
        sudo chmod 766 /etc/bash.bashrc
         echo "PKG_CONFIG_PATH=
$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig" >> /etc/bash.bashrc
         echo "export PKG_CONFIG_PATH" >> /etc/bash.bashrc
        sudo cp /usr/local/lib/python2.7/site-packages/cv.so
/usr/local/lib/python2.7/dist-packages/cv.so
########Instalacion de Java
        if ! hash java 2>&-; then
         sudo add-apt-repository
"deb http://archive.canonical.com/ lucid partner"
         sudo apt-get update
         sudo apt-get install
sun-java6-jre sun-java6-bin sun-java6-jdk
sun-java6-plugin sun-java6-fonts
        fi
########Compilacion e inicializacion del proyecto
        cd "$DLHOME"
        cd $PROYECTO
        javac ServidorImagen.java
        javac ClienteImagen.java
        g++ `pkg-config --cflags --libs opencv`
-o proyecto proyecto.cpp
        java ServidorImagen
```

6. Entrenamiento

En este apartado se explica la etapa 3. Para poder realizar la detección de objetos se entreno una red neuronal con la ayuda de la librería OpenCV. Los pasos que se siguieron son: Se descarga e instala OpenCV. Se obtiene una foto del objeto que se quiere reconocer en distintos ángulos o perfiles. Se obtienen imágenes donde NO se encuentre el objeto que se desea reconocer. Lo recomendado es tener aproximadamente 3000 o más imágenes de este tipo. Se crea un archivo donde se guarda la ruta de estas imágenes a las que les llamaremos imágenes negativas puesto que el objeto no se encuentra en ellas. Se obtienen otras imágenes donde tampoco este el objeto, pero con la diferencia de que a este paquete de imágenes le agregaremos la imagen del objeto que deseamos reconocer. Lo recomendado es tener 5000 o más imágenes de este tipo de referencia diferentes a las imágenes negativas. Se crea un archivo donde se guarda la ruta de estas imágenes a las que les llamaremos imágenes positivas. Se ejecuta una herramienta de OpenCV para crear las imágenes positivas(donde cada imagen contiene al objeto), desde una terminal ejecutamos:

Listing 20. Crear Muestras

```
opencv_createsamples -img /home/usuario/logo.jpg -num 5000
-bg /home/usuario/positivas.dat -info positivasfinal.dat -maxxangle
1.1 -maxyangle 1.1 -maxzangle 0.5 -maxidev 100 -w 20 -h 20 -bgcolor 0
-bgthresh 0
```

- -img: indica donde se encuentra la imagen del objeto a reconocer.
- -num: indica el numero de imágenes positivas que tenemos.
- -bg: indica el archivo donde están las rutas de nuestras imágenes positivas.
- -info: indica el archivo que se va a generar con datos acerca de la imagen original(sin el objeto) y en que posición se coloca el objeto dentro de esta imagen.
- maxxangle: indica que tanto se deforma el objeto en el eje x.
- maxyangle: indica que tanto se deforma el objeto en el eje y.
- maxzangle: indica que tanto se deforma el objeto en el eje z.
- -w 20 -h 20: indican el tamaño de la muestra.
- -bgcolor: indica el color de fondo, en este caso las muestras serán todas en blanco y negro.

Ya que se genero el archivo positivasfinal.dat se convierte a un archivo *.vec, desde una terminal ejecutamos:

Listing 21. Archivo Vector

```
opencv_createsamples -info positivasfinal.dat
-vec positivasfinal.vec -num 5000 -w 20 -h 20
```

Por último ejecutamos desde una terminal el siguiente comando el cual mediante del algoritmo haartraining[4] entrenará la red neuronal que utilizaremos para la detección de objetos en las imágenes:

Listing 22. Entrenamiento

```
opencv_haartraining -data nombre/de/carpeta/final
-vec positivasfinal.vec -bg negativas.dat -nstages 20
-nsplits 2 -minhitrate 0.998 -maxfalsealarm 0.5
-npos 5000 -nneg 2000 -w 20 -h 20 -nonsym
-mem 2048 -mode ALL
```

Este último comando ejecuta el algoritmo de entrenamiento haar-training.

- -data: indica la carpeta donde se guardaran los resultados del entrenamiento.
- -vec: indica nuestro archivo *.vec generado anteriormente.
- -bg: indica el archivo de rutas de las imágenes negativas.
- -nstages: indica el número de etapas que abarcara el entrenamiento, en este caso 20 son optimas.
- -minhitrate 0.998 -maxfalsealarm 0.5: indican la tolerancia si se llega a ese punto entonces es cuando el entrenamiento termina.
- -npos 5000 -nneg 2000: indican el numero de imágenes positivas y negativas respectivamente.
- -mem 2048: indica la cantidad máxima de memoria RAM que se puede utilizar.

Cuando se alcanza la tolerancia requerida el entrenamiento terminará y se generara un archivo *.xml en la carpeta que especificamos anteriormente a este archivo se le llama cascada. Sino se crea el archiv *.xml pero ya termino el entrenamiento, o si aún no se termina el entrenamiento pero se quiere obtener el archivo *.xml entonces desde la terminal accedemos a la carpeta */OpenCV/bin* y ejecutamos:

Listing 23. Convertir Cascada

```
./convert_cascade --size="20x20" nombre/de/carpeta/final
nombrefinaldelarchivo.xml
```

Siempre *size* tiene que coincidir con -w y -h.

Los archivos *.xml generados por el entrenamiento son utilizados por el programa Proyecto.cpp por lo tanto siempre deben de ir en la carpeta llamada Entrenamiento. Para agregar una cascada nueva a Proyecto.cpp se tiene que crear una nueva función Objeto(), donde cargue los parámetros de esta nueva cascada.

7. Pruebas e Integración del Proyecto

En este apartado se explica la etapa 4.
Las pruebas se fueron realizando en cada etapa del desarrollo. La primer prueba fue confirmar la recepción de la imagen tomada desde el dispositivo móvil al servidor. La cual se describe en la figura 8 y 9.

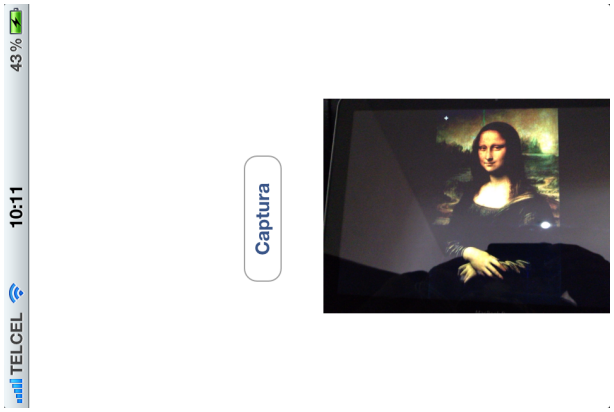

Figura 8. La foto siendo tomanda desde el dispositivo móvil.

Figura 9. La foto siendo recibida en el servidor.

La segunda prueba se realizó una vez que se tenia la imagen en el servidor. Ejecutar, desde el programa ServidorImagen.java, el programa Proyecto.cpp esto se logró implementando los métodos:

Listing 24. ServidorImagen.java

```
File wd = new File("/home/"+usuario+"/PT");
        System.out.println(wd + "\n");
```

```
        Process proc = null;
        String comando[] = {"gnome-terminal","-e","./Proyecto"};
try {
        proc = Runtime.getRuntime().exec(comando, null, wd);
        System.out.println(proc);
}
catch (IOException e) {
        e.printStackTrace();
}
```

Estos métodos crean un espacio de trabajo que en este caso es */home/usuario/PT/Imagenes* aqui es donde se encuentra el proyecto, ya que se creo el espacio de trabajo, se crea el comando que se va a ejecutar en este caso es: gnome-terminal el cual ejecuta una terminal de linux, -e para ejecutar el siguiente parámetro que es ./Proyecto. La Figura 10 describe este proceso.

Figura 10. Se puede verificar como se manda imprimir el espacio de trabajo */home/usuario/PT/Imagenes* y se ve la ejecución de un subproceso POSIX .

La tercer prueba fue verificar si se estaban cargando correctamente las cascadas y si se estaban detectando los objetos para los que estaban entrenadas. la figura 11 y 12 muestran este proceso.

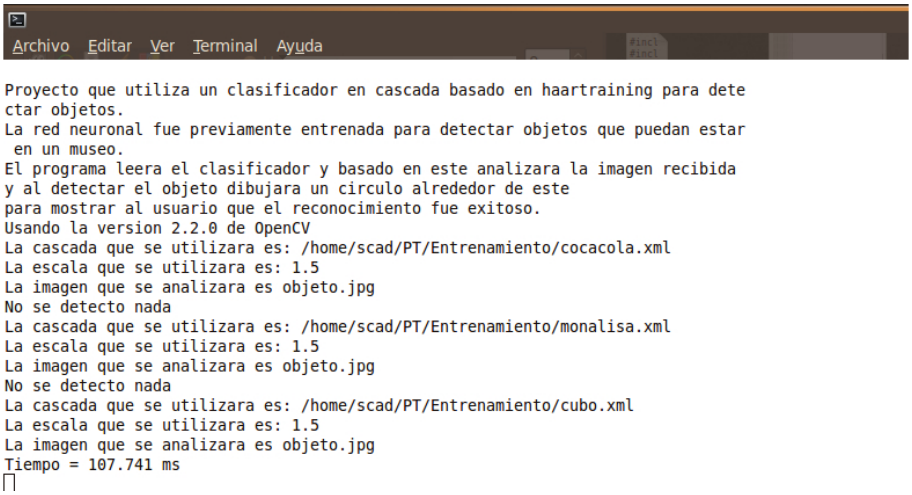

Figura 11. Se muestra como se cargan las cascadas pero como la foto no contiene ningún objeto conocido no se detecta nada.

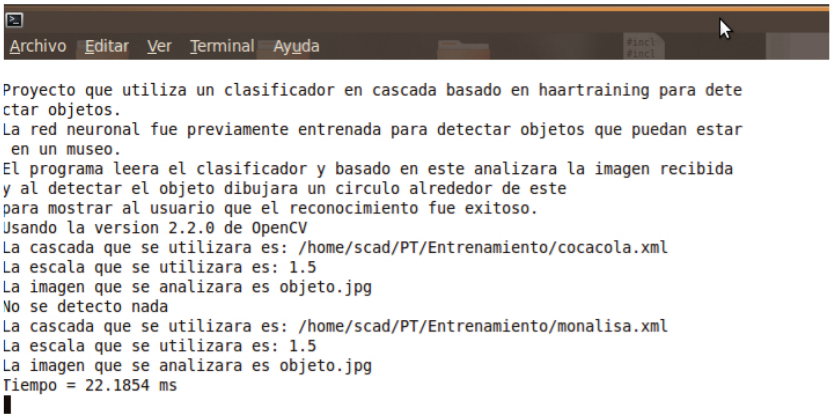

Figura 12. Se muestra como se carga la primer cascada y no detecta nada con ella, seguido de la segunda cascada la cual detecta el objeto para la cual fue entrenada.

La cuarta prueba es verificar que la imagen después de ser analizada se regrese al dispositivo móvil esto se logró ejecutando desde el programa Proyecto.cpp la siguiente función:

Listing 25. Proyecto.cpp

```
system("java ClienteImagen");
```

La cual ejecuta el programa ClienteImagen.java la figura 13 y 14 muestran este proceso.

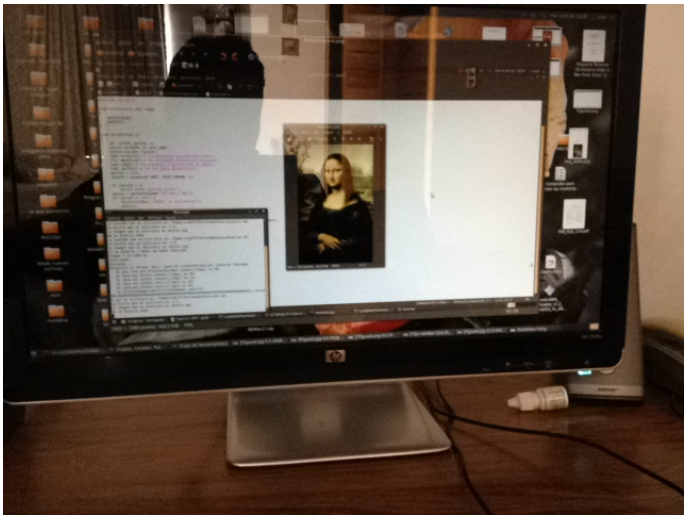

Figura 13. Muestra la foto original que llega al servidor.

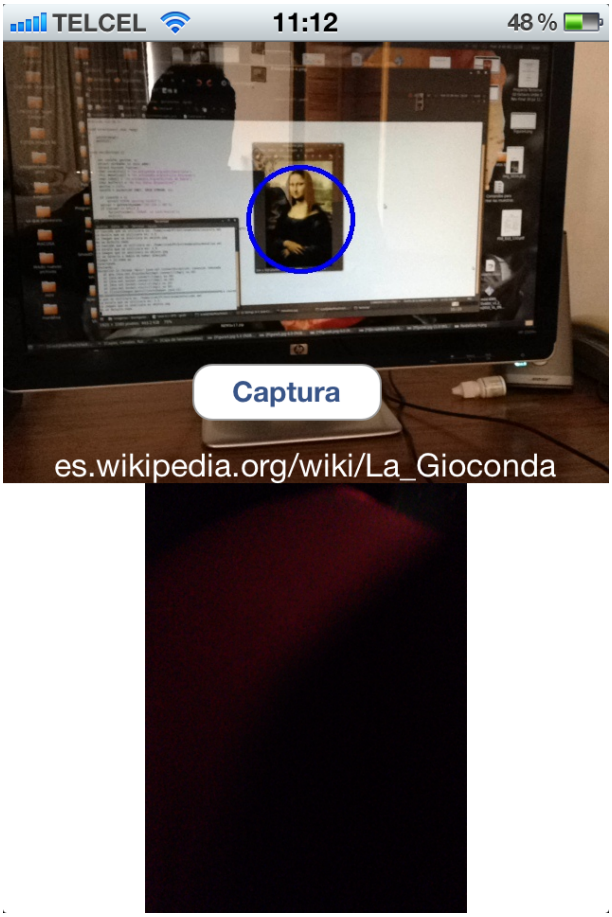

Figura 14. Muestra la foto ya analizada y como detecto un objeto la marco y mando información acerca de ese objeto.

8. Apéndice A

8.1. OpenCV. Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones en control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.
Para el desarrollo de este proyecto se utilizó OpenCV debido a sus capacidades para el reconocimiento de objetos su facilidad de uso y su perfecta adaptación al proyecto debido a esto el sistema desarrollado es multiplataforma, puesto que fue probado en sistemas bajo el sistema operativo Ubuntu, Mac OSX Lion, Debian y Windows.

OpenCV se divide en 5 grupos:

- CXCORE: donde se encuentran las estructuras y algoritmos básicos que usan las demás funciones.
- CV: donde están implementadas las funciones principales de procesamiento de imágenes.
- HighGUI: todo lo relacionado a la interfaz gráfica de OpenCV y las funciones que permiten importar imágenes y video.
- ML: que cuenta con algoritmos de aprendizaje, clasificadores y demás.
- CvAux: con funciones experimentales.

9. Apéndice B

9.1. Código Fuente. En esta sección se pondrá el código fuente por completo.

Listing 26. PTVideoViewController.h

```
#import <UIKit/UIKit.h>
#import <CoreVideo/CoreVideo.h>
#import <CoreMedia/CoreMedia.h>
#import <AVFoundation/AVFoundation.h>
#import <ImageIO/ImageIO.h>
#import <QuartzCore/QuartzCore.h>



@interface PT_VideoViewController : UIViewController<NSStreamDelegate>

    UIView *vistaSobrepuesta;
    UILabel *Informacion;
    AVCaptureStillImageOutput *ImagenSalida;
    UIImageView *vImage;
}
@property (nonatomic, retain) IBOutlet UIView *vistaSobrepuesta;
@property (nonatomic, retain) IBOutlet UILabel *Informacion;
@property (nonatomic, retain) AVCaptureStillImageOutput *ImagenSalida;
@property (nonatomic, retain) IBOutlet UIImageView *vImage;

//Declaracion de metodos
-(IBAction)Captura;
-(void) iniComuSmall:(NSData*)imagen;
-(void) iniServidor;
-(void) iniServidorInformacion;

@end
```

Listing 27. PTVideoViewController.m

```
#import "PT_VideoViewController.h"
#import <CoreVideo/CoreVideo.h>
#import <CoreMedia/CoreMedia.h>
```

```
#import <AVFoundation/AVFoundation.h>
#import <ImageIO/ImageIO.h>
#import <QuartzCore/QuartzCore.h>
#import "AbstractSocket.h"
#import "Socket.h"

@implementation PT_VideoViewController
@synthesize vistaSobrepuesta;
@synthesize Informacion;
@synthesize ImagenSalida;
@synthesize vImage;

//Se crea una nueva propiedad

-(void) viewDidAppear:(BOOL)animated{

UIButton *boton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[boton addTarget:self action:@selector(Captura)
forControlEvents:UIControlEventTouchDown];
[boton setTitle:@"Captura" forState:UIControlStateNormal];
boton.frame = CGRectMake(100, 170, 100, 30);
[self.view addSubview:boton];


AVCaptureSession *sesion = [[AVCaptureSession alloc] init];
sesion.sessionPreset =
    AVCaptureSessionPreset640x480;
    //AVCaptureSessionPreset1280x720;
    //AVCaptureSessionPresetHigh;
    //AVCaptureSessionPresetMedium;

        CALayer *CapaVista = self.vistaSobrepuesta.layer;
        NSLog(@"viewLayer = %@",CapaVista);

      AVCaptureVideoPreviewLayer *VistaPreviaVideo =
      [[AVCaptureVideoPreviewLayer alloc] initWithSession:sesion];

      VistaPreviaVideo.frame = self.vistaSobrepuesta.bounds;
```

```
        [self.vistaSobrepuesta.layer addSublayer:VistaPreviaVideo];

       AVCaptureDevice *dispositivo =
    [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

         NSError *error = nil;

     AVCaptureDeviceInput *entrada = [AVCaptureDeviceInput
           deviceInputWithDevice:dispositivo error:&error];
         if(!entrada) {
            NSLog(@"Error Tratando de abrir la camara: %@",error);
         }

         [sesion addInput:entrada];

     //Variables para el formato de video que se va a mostrar
         ImagenSalida = [[AVCaptureStillImageOutput alloc] init];
         NSDictionary *Salidas = [[NSDictionary alloc]
initWithObjectsAndKeys: AVVideoCodecJPEG, AVVideoCodecKey, nil];
         [ImagenSalida setOutputSettings:Salidas];

         [sesion addOutput:ImagenSalida];

         [sesion startRunning];

         [super viewDidAppear:YES];
}

-(IBAction) Captura{
         AVCaptureConnection *ConexVideo = nil;
for (AVCaptureConnection *conexion in ImagenSalida.connections){
       for (AVCaptureInputPort *puerto in [conexion inputPorts]){
         if ([[puerto mediaType] isEqual:AVMediaTypeVideo] ){
                         ConexVideo = conexion;
                         break;
                         }
                 }
                 if (ConexVideo) { break; }
         }
```

```
        NSLog(@"Pedimento de captura de video: %@", ImagenSalida);
       [ImagenSalida captureStillImageAsynchronouslyFromConnection:
        ConexVideo completionHandler: ^(CMSampleBufferRef
                        imageSampleBuffer, NSError *error)
           {
        CFDictionaryRef exifAttachments = CMGetAttachment
       ( imageSampleBuffer, kCGImagePropertyExifDictionary, NULL);
                  if (exifAttachments){
                          NSLog(@"Datos: %@", exifAttachments);
                  }
                  else
                          NSLog(@"Sin Datos");

          NSData *DatosImagen =
[AVCaptureStillImageOutput jpegStillImageNSDataRepresentation:
                               imageSampleBuffer];


           [self iniComuSmall:DatosImagen];
           [self iniServidor];
           [self iniServidorInformacion];
           }];
}


- (void) iniComuSmall:(NSData *)imagen{
    Socket *socket;
    int port = 3499;
    NSString *host = @"192.168.1.77";

    socket = [Socket socket];

    @try {
        [socket connectToHostName:host port:port];
        [socket writeData:imagen];
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
```

```
    }

    [socket close];

}


- (void)iniServidor{

    Socket *socket;
    NSMutableData *respuesta = [[[NSMutableData alloc]
                                    init] autorelease];
    NSString *resultado;

    socket =[[Socket alloc] init];
    [socket listenOnPort:3498];

    @try {
        [socket acceptConnection];
        while ([socket readData:respuesta]) {
        NSLog(@"Recibiendo Imagen");
        }
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
    }
    [socket close];
    resultado = [[[NSString alloc] initWithData:respuesta
                encoding:[NSString defaultCStringEncoding]]
                                                autorelease];
    UIImage *imagen = [[UIImage alloc]
                initWithData:respuesta];
    self.vImage.image = imagen;

}
- (void)iniServidorInformacion{

    Socket *socket;
```

```
    NSMutableData *respuesta =
     [[[NSMutableData alloc] init] autorelease];
    NSString *resultado;
    socket =[[Socket alloc] init];
    [socket listenOnPort:3497];

    @try {
        [socket acceptConnection];
        while ([socket  readData:respuesta]) {
            NSLog(@"Recibiendo Informacion");
        }
    }
    @catch (NSException *exception) {
        NSLog(@"Error: %@", exception);
        socket = nil;
    }
    [socket close];
    resultado = [[NSString alloc]
initWithData:respuesta encoding:NSASCIIStringEncoding];
    Informacion.text = resultado;
}
- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
}
- (void)viewDidUnload {
    self.vistaSobrepuesta = nil;
}
- (void)dealloc {
    [PT_VideoViewController release];
    [ImagenSalida release];
    [vImage release];
    [vistaSobrepuesta release];
    [Informacion release];
    [super dealloc];
}
@end
```

Listing 28. ServidorImagen.java

```
import java.net.*;
import java.io.*;

public class ServidorImagen {
  public static void main (String [] args )
                        throws IOException {

    ServerSocket servsock = new ServerSocket(3499);
    long inicio = System.currentTimeMillis();
    int BytesLeidos;
    int actual = 0;
    int tamarch=6022386;
    String usuario = System.getProperty("user.name");

    while (true) {
      System.out.println("\nEsperando...");

      Socket sock = servsock.accept();
      System.out.println("Coneccion Aceptada : " + sock);

    byte [] ArregloBytes  = new byte [tamarch];
    InputStream is = sock.getInputStream();
    FileOutputStream fos = new
    FileOutputStream
    ("/home/"+usuario+"/PT/Imagenes/objeto.jpg");
    BufferedOutputStream bos =
    new BufferedOutputStream(fos);
    BytesLeidos =
    is.read(ArregloBytes,0,ArregloBytes.length);
    actual = BytesLeidos;

    do {
       BytesLeidos = is.read(ArregloBytes, actual,
                   (ArregloBytes.length-actual));
       if(BytesLeidos >= 0) actual += BytesLeidos;
    } while(BytesLeidos > -1);

    bos.write(ArregloBytes, 0 , actual);
```

```
    bos.flush();
    long fin = System.currentTimeMillis();
    System.out.println(fin-inicio);
    bos.close();
    sock.close();

        File wd = new File("/home/"+usuario+"/PT");
        System.out.println(wd + "\n");
        Process proc = null;
        String comando[] =
        {"gnome-terminal","-e","./Proyecto"};
    try {
        proc =
    Runtime.getRuntime().exec(comando, null, wd);
        System.out.println(proc);
    }
    catch (IOException e) {
        e.printStackTrace();
    }

    if (proc != null) {
        System.out.print(proc.getErrorStream());

    try {
        proc.waitFor();
        proc.destroy();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    }
    }

  }


}
```

Listing 29. Proyecto.cpp

```
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include "sock.h"

#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;
using namespace cv;

void Acerca();
int DetectaryDibujar(Mat& img, CascadeClassifier& cascad,
double scale, int msj);
int Objeto1();
int Objeto2();
int Objeto3();

int main(){
    int resultado = 0, i = 0;

    Acerca();
    resultado = Objeto1();
    if(!resultado == 0){
        system("java ClienteImagen");
        i = 1;
       recibir(i);
       exit(0);
    }
    resultado = Objeto2();
    if(!resultado == 0){
       system("java ClienteImagen");
       i = 2;
       recibir(i);
    }
    resultado = Objeto3();
    if(!resultado == 0){
```

```
        system("java ClienteImagen");
        i = 3;
        recibir(i);
    }
    if(resultado == 0){
        system("java ClienteImagen");
        i = 0;
        recibir(i);
    }

return 0;


}


int Objeto1(){

    Mat Imagen;
    int resultado = 0, mensaje = 0;
    String usuario = getenv("USER");
    String cascadaCocacola =
    "/home/"+usuario+"/PT/Entrenamiento/cocacola.xml";
    CascadeClassifier cascada;
    double escala = 1;

    cascada.load(cascadaCocacola);
    if(cascada.empty())
        cout << "La cascada no se pudo cargar" << endl;

    cout << "La cascada que se utilizara es: "
         << cascadaCocacola << endl;
    escala = 1.5;
    cout << "La escala que se utilizara es: "
         << escala << endl;
    Imagen = imread("Imagenes/objeto.jpg", 1);
    cout << "La imagen que se analizara es objeto.jpg"
         << endl;
```

```
    resultado = DetectaryDibujar
              (Imagen, cascada, escala, mensaje);

    if(resultado == 0)
        return 0;
    else
        return 1;
}
int Objeto2(){

    Mat Imagen;
    int resultado = 0, mensaje = 0;
    String usuario = getenv("USER");
    String cascadaMonalisa =
    "/home/"+usuario+"/PT/Entrenamiento/monalisa.xml";
    CascadeClassifier cascada;
    double escala = 1;

    cascada.load(cascadaMonalisa);
    if(cascada.empty())
        cout << "La cascada no se pudo cargar"
             << endl;

    cout << "La cascada que se utilizara es:
       " << cascadaMonalisa << endl;
    escala = 1.5;
    cout << "La escala que se utilizara es: "
         << escala << endl;
    Imagen = imread("Imagenes/objeto.jpg", 1);
    cout
       << "La imagen que se analizara es objeto.jpg"
       << endl;
    resultado = DetectaryDibujar
                  (Imagen, cascada, escala, mensaje);

    if(resultado == 0)
        return 0;
    else
        return 1;
```

```
}
int Objeto3(){

    Mat Imagen;
    int resultado = 0, mensaje = 1;
    String usuario = getenv("USER");
    String cascadaCubo =
      "/home/"+usuario+"/PT/Entrenamiento/cubo.xml";
    CascadeClassifier cascada;
    double escala = 1;

    cascada.load(cascadaCubo);
    if(cascada.empty())
        cout << "La cascada no se pudo cargar"
             << endl;

    cout << "La cascada que se utilizara es:
       " << cascadaCubo << endl;
    escala = 1.5;
    cout << "La escala que se utilizara es:
       " << escala << endl;
    Imagen = imread("Imagenes/objeto.jpg", 1);
    cout
       << "La imagen que se analizara es objeto.jpg"
       << endl;
    resultado = DetectaryDibujar
              (Imagen, cascada, escala, mensaje);

    if(resultado == 0)
        return 0;
    else
        return 1;
}


int DetectaryDibujar
(Mat& img,
CascadeClassifier& cascad, double scale, int msj)
{
```

```
    int i = 0, fallo = 0;
    double t = 0;
    vector<Rect> caras;
    const static Scalar colores[] =
{CV_RGB(0,0,255), CV_RGB(0,128,255),
CV_RGB(0,255,255), CV_RGB(0,255,0),
CV_RGB(255,128,0), CV_RGB(255,255,0),
CV_RGB(255,0,0), CV_RGB(255,0,255)};
    Mat gris, smallImg(cvRound
(img.rows/scale), cvRound(img.cols/scale),
CV_8UC1);

    cvtColor(img, gris, CV_BGR2GRAY);
    resize(gris, smallImg, smallImg.size(),
0, 0, INTER_LINEAR);
    equalizeHist(smallImg, smallImg);

    t = (double)cvGetTickCount();
    cascad.detectMultiScale( smallImg,
caras, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE,
Size(30, 30) );
    if(caras.empty()){
        cout << "No se detecto nada" << endl;
        fallo = 0;
        if(msj == 1){
            cv::imwrite
              ("Imagenes/resultado.jpg", img);
            system("java ClienteImagen");
            return fallo;
        }
        else{
            return fallo;
        }
        }
    else {
        fallo = 1;
    t = (double)cvGetTickCount() - t;
    printf("Tiempo = %g ms\n",
    t/((double)cvGetTickFrequency()*1000.));
```

```
    for(vector<Rect>::const_iterator r =
     caras.begin(); r != caras.end(); r++, i++)
    {
        Point centro;
        Scalar color = colores[i%8];
        int radio;
        centro.x = cvRound
((r->x + r->width*0.5)*scale);
        centro.y = cvRound
((r->y + r->height*0.5)*scale);
        radio = cvRound
((r->width + r->height)*0.25*scale);
        circle(img,
centro, radio, color, 3, 8, 0);
    }

    cv::imwrite("Imagenes/resultado.jpg", img);
    system("java ClienteImagen");
 }
    return fallo;
}

void Acerca()
{
    cout << "\nProyecto que utiliza un clasificador
 en cascada basado en haartraining para detectar
objetos.\n";
    cout << "La red neuronal fue previamente
entrenada para detectar objetos que puedan estar
en un museo.\n";
    cout << "El programa leera el clasificador y
basado en este analizara la imagen recibida y al
detectar el objeto dibujara un circulo alrededor
de este\n";
    cout << "para mostrar al usuario que el
reconocimiento fue exitoso.\n";
    cout << "Usando la version " << CV_VERSION
<< " de OpenCV"<< endl;
}
```

Listing 30. sock.h

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}

void recibir(int i)
{
    int sockfd, puerto, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char cocacola[] = "es.wikipedia.org/wiki/Coca-Cola";
    char monalisa[] = "es.wikipedia.org/wiki/La_Gioconda";
    char cubo[] = "es.wikipedia.org/wiki/Cubo_de_Rubik";
    char buffer[] = "No hay datos disponibles";
    puerto = 3497;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    if (sockfd < 0)
        error("Error al abrir el socket");
    server = gethostbyname("192.168.1.86");
    if (server == NULL) {
        fprintf(stderr,"Error host no valido\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
(char *)&serv_addr.sin_addr.s_addr, server->h_length);
```

```
    serv_addr.sin_port = htons(puerto);
    if (connect(sockfd,(struct sockaddr *)
&serv_addr,sizeof(serv_addr)) < 0)
        error("Error conectando");
    switch (i){
        case 1:
                n = write(sockfd,cocacola,strlen(cocacola));
                break;
        case 2:
                n = write(sockfd,monalisa,strlen(monalisa));
                break;
        case 3:
                n = write(sockfd,cubo,strlen(cubo));
                break;
        case 0:
                n = write(sockfd,buffer,strlen(buffer));
                break;
    }

    if (n < 0)
         error("Error al escribir en el socket");
    close(sockfd);
}
```

Listing 31. ClienteImagen.java

```
import java.net.*;
import java.io.*;

public class ClienteImagen{
  public static void main (String [] args ) throws IOException {

    // Conexion al iPhone por el puerto
    Socket sock = new Socket("192.168.1.86",3498);
    System.out.println("Conectando...");

      // Envio del archivo
      File Archivo = new File ("Imagenes/resultado.jpg");
      byte [] ArregloBytes  = new byte [(int)Archivo.length()];
      FileInputStream fis = new FileInputStream(Archivo);
      BufferedInputStream bis = new BufferedInputStream(fis);
      bis.read(ArregloBytes,0,ArregloBytes.length);
      OutputStream os = sock.getOutputStream();
      System.out.println("Enviando...");
      os.write(ArregloBytes,0,ArregloBytes.length);
      os.flush();
      sock.close();
      }

}
```

10. Apéndice C

10.1. Manual de Usuario. Instalación El sistema fue desarrollado con la finalidad de que pueda ser instalado en cualquier computadora corriendo Ubuntu 11.04 en adelante.

Los pasos para la instalación son:

- Copiar la carpeta PT en la ruta */home/usuario/* y el archivo instalacion.sh.
- Dar permisos de ejecución al script instalacion.sh — desde una terminal ejecutar: ˜$ chmod 777 instalacion.sh
- Ejecutar el script desde la carpeta */home/usuario/* — ˜$ ./instalacion.sh.

Después de realizado esto el proyecto quedará instalado y corriendo en su totalidad.

El siguiente paso es instalar el cliente en el dispositivo iOS puedes ser un iPhone, iPod Touch o un iPad[7] la versión del sistema operativo que tiene que tener el dispositivo es iOS 5.0 en adelante, se tiene que contar con un computadora Mac con Xcode[8] 4 en adelante para poder cargar el proyecto. También se debe de contar con una licencia de desarrollador[5] para poder ejecutar el proyecto en el dispositivo. Si se cumple con los requisitos, solo se tiene que conectar el dispositivo a la computadora, cargar el proyecto con Xcode y ejecutarlo en el dispositivo.

[7]Sistema operativo exclusivo de los dispositivos: iPhone, iPad y iPodTouch.

[8]IDE de desarrollo gratuito, para desarrollo aplicaciones basadas en Objective-C y C/C++ creado por Apple Inc.

Referencias

[1] J. Vallino, *Introduction to Augmented Reality*, Augmented Reality, 2006.[En Línea]. Disponible: http://www.se.rit.edu/ jrv/research/ar/introduction.html [consultado 2011 diciembre 5].

[2] S. Frank, *A lightweight Objective-C wrapper for BSD sockets on Mac OS X.*, SmallSockets, 2001.[En Línea]. Disponible: http://sourceforge.net/projects/smallsockets/ [consultado 2011 diciembre 5].

[3] OpenCV, *OpenCV Guide*,OpenCV, 2008. [En Línea]. Disponible: http://cgi.cse.unsw.edu.au/ cs4411/wiki/index.php?title=OpenCV$_{G}uide[consultado2011diciembre5]$.

[4] D.J. Barnes, *Quotient Robotics — Hobby Robotics, Biped Robots C Robotics*, Quotient Robotics, 2008. [En Línea]. Disponible: http://www.quotientrobotics.com/2010/04/opencv-haartraining-object-detection$_{0}9.html[consultado2011diciembre5]$.

[5] Apple, *iOS Developer Program*, Apple, 2008. [En Línea]. Disponible: http://developer.apple.com/programs/ios/ [consultado 2011 diciembre 5].