telenor

R&I Research Note

R&I N 22/2008

Espen Egeland, Sigmund Akselsen, Bente Evjemo and Anders Schürmann

An image search client for the Apple iPhone

R&I Research Note	**N 22/2008**
Title	**An image search client for the Apple iPhone**
Author(s)	Espen Egeland, Sigmund Akselsen, Bente Evjemo and Anders Schürmann
ISBN / ISSN	/0809-1021
Security group	OPEN
Date	2008.10.20

Abstract
This document describes the development of a visual search client for the Apple iPhone. The image recognition is based on Evolution Robotics' ViPR tool.

Keywords
image recognition, ViPR, iPhone

Preface

Image recognition is fundamental to a wide range of emerging mobile services and Telenor R&I has through the VISEP – Visual Search Pilot, CAIM – Context-Aware Image Management (partly financed through the Norwegian Research Council) and SAPIR - Search In Audio Visual Content Using Peer-to-peer IR (partly financed through the European Union) projects paid special attention to the exploration of mobile services dependent of effective image recognition.

In March 2008 a licence of ViPR (Visual Pattern Recognition) - a promising image recognition software provided by Evolution Robotics, was made available for piloting new services. This document describes the development of an image search client for the Apple iPhone that communicates with a server running the ViPR software.

The study is a part of the CAIM-project and has been financed through funds supporting cooperation between Telenor and the University of Tromsø. The work has been conducted by Espen Egeland, under the supervision of Anders Schürmann, Bente Evjemo and Sigmund Akselsen.

Tromsø, October 2008.

Sigmund Akselsen, Bente Evjemo and Anders Schürmann

Contents

1 Introduction

Evolution Robotics' ViPR[1] (visual pattern recognition) software has the ability to detect and recognize complex visual patterns. It can be used in various applications:

- Interfaces to the internet on mobile devices (PDAs, cell phones, etc.)
- Visual search engine
- Navigation systems for vacuums, UAVs, etc.
- Security systems for retail stores, airports, etc.

Telenor R&I has acquired a licence of the ViPR software and uses it for experimenting with various mobile search applications. As part of the VISEP Visual Search Pilot project the server part of the software has been installed and an image search client for the Nokia N95 has been developed[2].

In parallel an image search client for the Apple iPhone has been developed by the CAIM project. The motivation for taking on this development has been to explore the iPhone features for making appealing GUIs to mobile applications, among other touch gestures for easy manipulation of images. Also it has been a goal to get some practical experiences with developing client-server software solutions for the iPhone in general.

This document outlines how the image search client for the Apple iPhone is constructed. The client is programmed in Object C language and communicates with the ViPR software. It was developed using Xcode 3.1 development environment on a Macbook Pro laptop. The iPhone SDK, including Xcode can be downloaded free from Apple Developer.

[1] *http://www.evolution.com/core/ViPR/*
[2] *http://playground.telenor.com/node/789*

2 Model overview of the GUI classes

The blue boxes in figure 1 show classes that are defined in the program, while the red boxes represent standard classes that have not been sub classed. The classes marked NIB, include graphical elements such as buttons, pictures, etc. that are defined in a corresponding NIB file. A NIB file is made using the Interface Builder, a graphical tool for constructing user interfaces included with the iPhone SDK. Button presses in a view are handled in the corresponding view controllers.

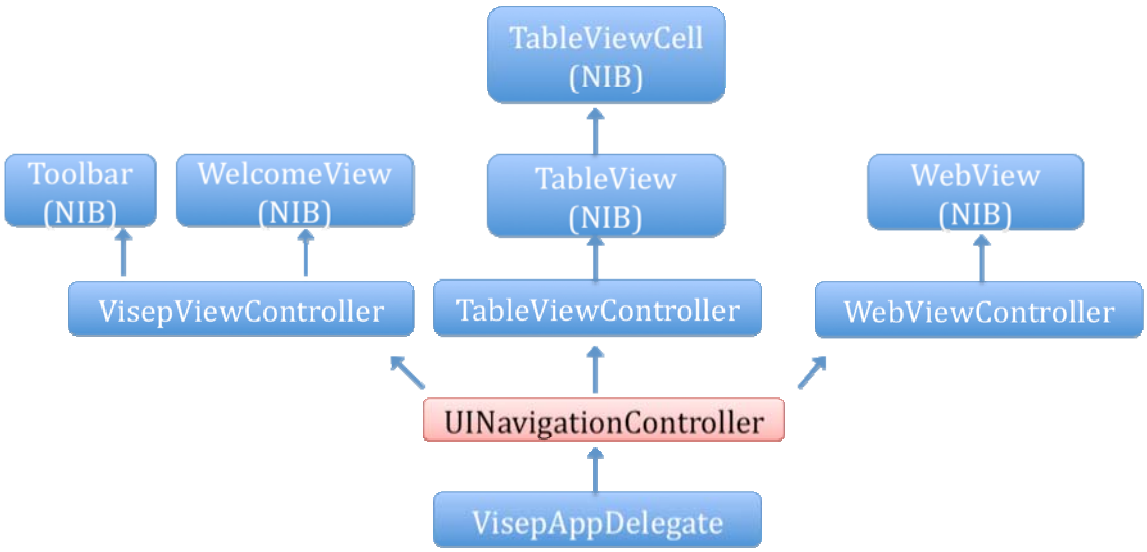

Figure 1. Classes that make up the user interface.

3 The complete data model

Figure 2 shows a model of all the classes that make up the application. We will briefly describe the most important ones. For details on the methods of each class please see the commented source code.

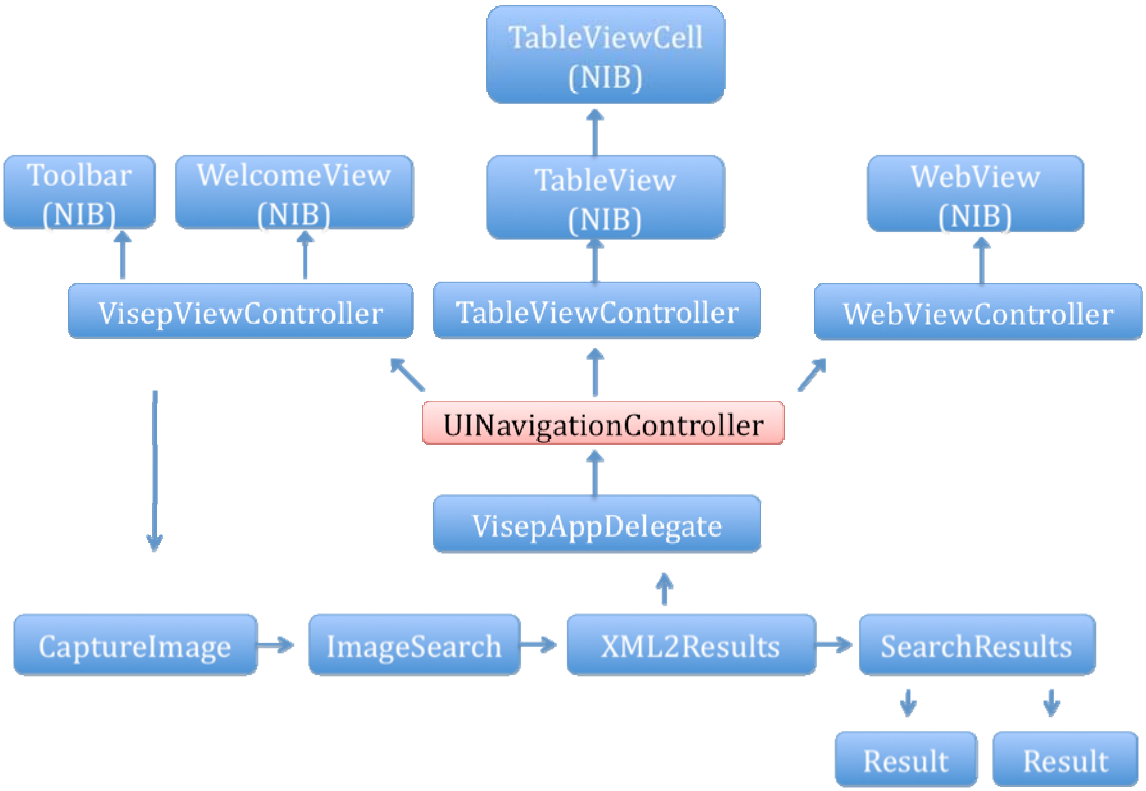

Figure 2: Classes of the applications.

3.1 VisepAppDelegate

This is the first class and it is created when the application is activated. It is responsible for setting up the navigation controller, and all view controllers. When starting an instance of the VisepViewController is created, which presents the user with the welcome screen. When the results of a query have been processed it creates a TableViewController to display the results. When a result is chosen by the user, it is also responsible for creating a WebView for displaying the resulting web page.

3.2 VisepViewController class

The VisepViewController handles keypresses for the welcome screen, and is also set as the delegate for the protocol defining use of the camera and gallery. It therefore implements delegate methods that are automatically called when an image has been picked, or the image picker is cancelled.

3.3 CaptureImage class

This class implements the standard methods for accessing the gallery and camera.

3.4 ImageSearch class

This class sets up an http connection with the server. It encodes the image and adds the necessary formatting of the http headers and body. It also receives the reply, and initializes an XML2Results object to parse the XML returned from the server.

3.5 XML2Results class

This class parses the xml returned from the server. For each result it receives it creates an object of type Result where it stores the different XML attributes. It also creates an object of type SearchResults as a container for all Results objects.

3.6 SearchResults class

This class contains an array of Results objects.

3.7 Result class

This class stores the parameters returned from the server for one successful hit in the image recognition system.

3.8 TableViewController

This class creates a tableview for displaying the results of the search. It uses a custom made ”cell” to display each result. The layout of the cell is defined in a NIB file called ResultCell.NIB, and its properties are defined in the class TableViewCell.

3.9 WebViewController

This class is the controller class for the WebView that shows the resulting webpage from a search. It handles input from the WebView toolbar (back, forward, reload buttons). It is also the delegate of the WebView, so that whenever a new webpage is loaded it is notified. This is used to automatically display and hide the toolbar.

4 User interface and usage

After installing the client on an iPhone an icon representing the image search client will be available on the phone's desktop. Pressing the icon will start the client.

The user is first presented with the screen shown in figure 3. The screen contains 2 buttons representing the available user actions. By pressing "Gallery" the phone will enter the photo gallery of the phone, and the user can select an image for use in the image search.

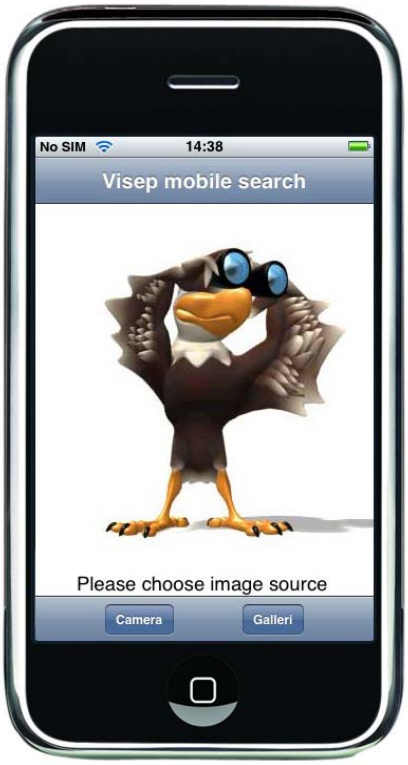

Figure 3: Welcome-screen of the mobile search client.

Alternatively, the user can press the "Camera"-button, and the phone's camera application will then launch in order for the user to take an image to be used in the query. The user can resize and section the image before submitting it in a query (as shown in figure 4a).

During image search, when data is transferred over the communication network, the client indicates that it is busy by displaying an animated icon. Upon receiving the result the screen will change and display the results in a list. If no hits for the image were identified the user will be informed of this by a popup-message. By clicking on a result in the result list the client will load an associated link into the client's embedded browser (as shown in figure 4b). From this screen the user can navigate the displayed web-page, launch the page in the iPhone's default Safari-browser, or return to the result overview page.

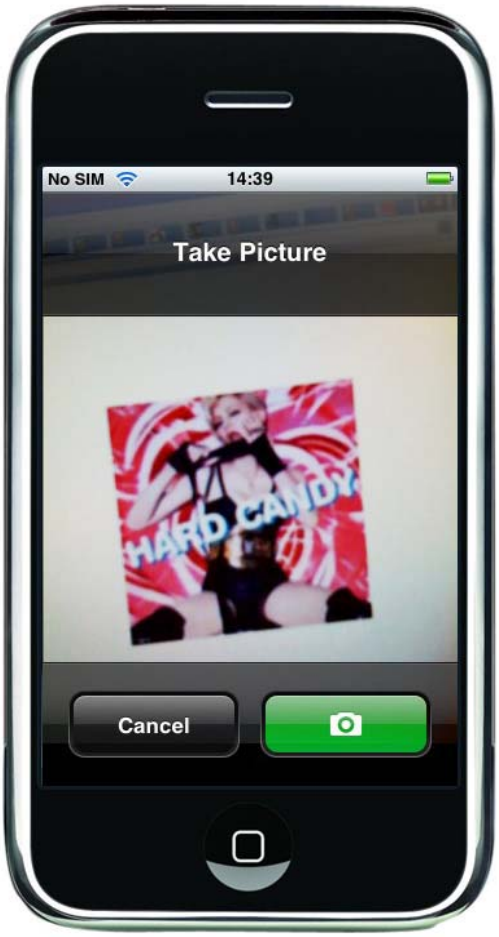

Figure 4a: Image taken of a CD-cover by the mobile search client.

Figure 4b: The result displayed for the search for the CD-cover.

5 Installation

If the iPhone has original non modified firmware, the application will have to be installed either through Apples AppStore, or using the Apples Ad Hoc method of distribution. Both of these methods require that the application is signed and compiled using a valid developer's certificate. In the case of Ad Hoc distribution, the device numbers of the phones where the application will run will need to be specified during compilation time.

If the iPhone has a modified/hacked firmware (2.0 or higher) that allows it to run unsigned code, the application can be installed as follows:

1. Make sure the phone has openSSH installed. If not, install it through Cydia or Installer. Use the root user, default password is "alpine".

2. Install Link Identity Editor (ldid) on the iPhone from Cydia or Installer. Ldid is a tool that will generate a valid hash signature for the application.

3. Use WinSCP or similar programs to copy the compiled file from the computer folder "project home directory"/build/releaseiphoneos/"Application name" to the iPhone folder /Applications.

4. SSH to the phone so that you have a command line interface. Navigate to the application folder and run ldid –S "application name" to generate a valid hash.

5. Restart springboard by either using a tool such as Bosspreps or by rebooting the phone.

6. The icon should now be visible on the springboard. Press to start. If the program crashes immediately after pressing the icon, the program was not hash signed properly. Repeat from step 4.

6 Testing and further work

The application has not been thoroughly tested, but has been installed and run successfully on a first generation Apple iPhone with OS version 2.0. The testing has been done over Wifi and Edge. We have not been able to test it with a 3G iPhone.

Since we had limited knowledge of what is in the ViPR database, we have been unable to test how it handles the situation when multiple results are received from a search. It should work fine, but it is untested.

The application is currently not multi-threaded. This means the user interface is not responsive while the network is accessed. There may also be a slight performance gain if it were multi-threaded.

The picture used on the welcome screen is not public domain, and should not be used for public demonstrations.