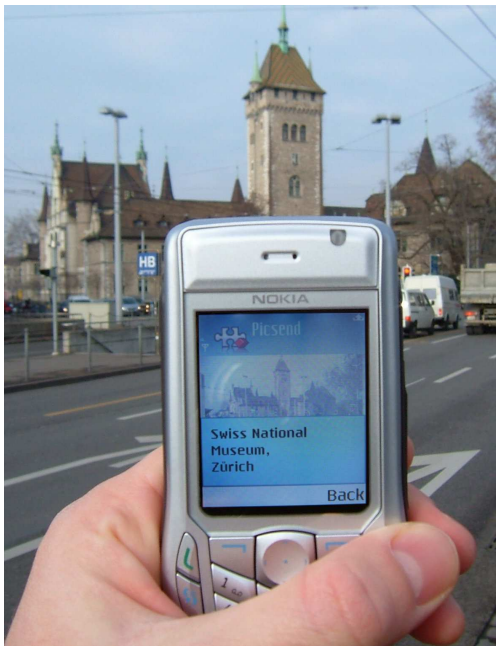
Object Recognition with Mobile Phones



Marcel Germann

Semester Thesis
Winter Semester 2005/2006

Computer Vision Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich
Prof. Luc Van Gool
Till Quack

Contents

Abstract

Today, most mobile phones have built-in digital cameras with a quality that improves with every new model. This semester thesis combines the use of mobile phone cameras with object recognition. In two approaches it shows that this is possible and applicable. In a first approach the images taken with the mobile phone camera are sent to a server which executes the expensive object recognition algorithms and the matching with a database. As communication method the Multimedia Message System (MMS) is used from the mobile phone to the server and the Short Message System (SMS) to send the answer back to the mobile phone. The result is a precise and extendable framework called *mobileor*. Applications could be price-comparison-systems or advertisement campaigns, where a user would be provided with more information about a product after sending a picture of it or of the advertisement placard. Also a city guide is possible where one just has to photograph an important building to get more information about it.
As a second approach the object recognition algorithm was completely ported to Symbian and runs server independent on a mobile phone. Even though the memory and processor limitations on a mobile phone are very restrictive, the resulting application is able to recognize objects.

Zusammenfassung

Heutzutage haben die meisten Mobilfunktelefone eingebaute Digitalkameras, die mit jedem Modell bessere Qualität haben. Diese Semesterarbeit kombiniert die Kameras von Mobilfunktelefonen mit Objekterkennung. In zwei Varianten wird gezeigt, dass dies möglich und anwendbar ist. In einer ersten Variante werden die mit der Kamera des Mobilfunkgerätes aufgenommenen Bilder an einen Server geschickt, der die aufwendigen Objekterkennungs-Algorithmen sowie die Vergleiche mit einer Referenzdatenbank übernimmt. Als Kommunikationsmethode vom Mobilfunktelefon zum Server wurde das Multimedia Message System (MMS) gewählt. Die Antwort zurück ans Mobilfunktelefon wird mittels Short Message System (SMS) gesendet. Das Resultat ist ein präzises und erweiterbares Framework genannt *mobileor*. Anwendungsmöglichkeiten könnten Preisvergleichungs-Systeme oder Werbekampagnen sein, wo ein Benutzer weitere Informationen über ein Produkt erhalten würde, wenn er ein Bild davon oder des Werbeplakates übermitteln würde. Auch wäre ein Stadtführer möglich, wo man lediglich ein wichtiges Gebäude fotografieren müsste, um genauere Informationen darüber zu erhalten.
Als zweite Variante wurde der Objekterkennungs-Algorithmus komplett auf Symbian portiert und läuft nun serverunabhängig auf einem Mobilfunktelefon. Obwohl die Speicher- und Prozessorkapazitäten eines Mobilfunktelefones sehr klein sind, ist die resultierende Applikation fähig ein Objekt zu erkennen.

Acknowledgements

I would like to thank Prof. Luc Van Gool and Till Quack to let me work in this project. The interesting idea behind it and the novelty on such a system have been a very strong motivation. Special thanks goes to Till for all the time and support he gave to me. He helped me through troubles and provided me with creative ideas.

Many thanks to my friends and family for always supporting me.

1 Introduction

This section gives a brief overview over mobile technologies used for this project, followed by the motivation and idea of this semester thesis.

1.1 Mobile Phone Technologies

Besides the normal telephone function of a mobile phone, devices nowadays offer much more technologies and applications. In this project, the ones for data transmission and of course the camera functions have been used. They will be described here briefly.

1.1.1 SMS

The manufacturer and provider created a messaging system, which gives them the possibility to exchange system messages or send information to their customers. A message can contain at most 160 characters. This short messaging service (SMS) was also offered to the end user. Especially in Europe where the rates for mobile telephone calls have been - at least at the beginning of this technology - very high, SMS became very popular. People got used to it and nowadays SMS is a widely used way to communicate. In Switzerland in 2005/2006 only on New Year's eve and New Year's Day 83.6 million SMS have been sent [NZZ06]. So with 6 million mobile phones [TA206], this is around 14 SMS per user.

1.1.2 MMS

In response to the success of SMS, the manufacturers and providers introduced an enhanced version called Multimedia Message System (MMS). This type of messages allows larger text content and file attachments. Because it is more complicated in handling (you have to enable MMS first on your mobile phone) and pricing, MMS had not that immense impact as SMS. But also the number of sent MMS grows every year.

1.1.3 Bluetooth and infrared

Besides provider based data transformation like SMS, MMS, and mobile Internet access, which all involve direct costs to the user, there are also provider independent communication possibilities. Today, the most common are infrared and bluetooth. The latter one is more and more replacing infrared. Bluetooth can be used very easily to send files from one device to another one or to a computer.

1.1.4 Cameras

Most of the newer mobile phones come with built-in photo cameras. Started as a "gadget" they now become better and reach a quality that is really usable for taking

good pictures. For this project, a mobile phone with a 1.3 mexapixel built-in camera was used.

1.2 Object Recognition with Mobile Phones

The Idea of this semester thesis is to use built-in cameras of mobile phones for object recognition. There are many algorithms for for detecting given objects on an image and this work shows that they also can be used in spite of the sometimes very bad quality of mobile phone images. To gain more speed and therefore be able to use a large database of objects to compare, the object recognition algorithm can be run on a server. The image taken on the mobile phone will be sent to this server which does the expensive calculation and sends back the result to the users mobile phone. This way it is possible to provide a large database for several users or also be able to gather information from the Internet. In this semester thesis a framework called *mobileor* (from ”mobile object recognition”) was developed, which offers the image handling, the object matching with a database and the communication with the mobile phone.

1.2.1 Applications

The mobileor framework offers with its functionality a wide field of possible applications. Here, some ideas are mentioned.

To give advertisements more interactivity, one could use this framework to provide more information about an ad. The user sends a picture of the placard or magazine page taken with the mobile phone camera and receives more information directly on the mobile phone. This could be detailed product information or the user takes part in a contest on this way and receives back the result. Also a direct order (with confirmation) of the photographed product for registered customers could be possible.

Another application would be a museum guide. You take a picture of the object and receive information about it. There had been done a lot of work to recognize bar codes with mobile phones (e.g. [Roh05] or [RW05]). These systems require such bar code tags on each object. For a museums guide this is applicable. But with the mobileor framework you are independent of bar codes or similar tags since it only uses the picture of the object itself. Therefore it would be possible to use it e.g. for a city guide. Tourists just take a photograph of e.g. a old church and the system gives them the name and more facts about the building (For an impression, see the image on the title page). No tags on the buildings would be needed. Of course the returned information could also be a city map with the actual position of the tourist marked.

Mobileor could also be used for stores or independent product comparison services. To give an example: You walk into a book store and you want to by a book. Before you do that, you take your mobile phone and send a picture to a comparison service. A few

seconds later you receive a list of bookstores and their prices for this book. This allows you to compare prices before you buy something.

As a last example, assistance for blind people should be mentioned. Object recognition on a mobile phone would give a low-cost help to distinct objects which are not different in shape but in texture (e.g. same shape but not the same color). Also orientation in larger environment, e.g. in a city, would be possible. Of course, the current developed system should then be extended with a audio output.

2 Concept

In this section the object recognition algorithm used for the mobileor framework is described, followed by the concept of two different approaches of mobileor. The first approach is a combination of a mobile phone and a server. The second setup is a completely independent on the mobile phone running system to learn objects and recognize them.

2.1 Local Features

There are many object recognition algorithms available. For this project it was important to use one which is robust against the sometimes bad quality of mobile phone pictures, as well as small changes in viewpoint. Local photometric descriptors have proved to be very successful for matching and object recognition ([Low04], [MS01], [SZ02], [SM97]). They work according to the following principle: first, the feature detection takes place, where interesting points in the picture are marked. Then, for each of those points a local descriptor is calculated, which depends only on the point and the area around it.

Mikolajczyk and Schmid [MS03] compared different local descriptors. They have shown that the by Lowe [Low04] proposed scale invariant feature transformation (SIFT) perform best. SIFT descriptors are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. Therefore the SIFT descriptors haven been chosen for this semester thesis. They are also especially useful because they are highly distinctive. You can compare the features of an image with a large database of features from many pictures and nevertheless reach a high probability for a correct match.

The scale invariant feature transform consists of the following major stages of computation ([Low04]):

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orien-

tation, scale, and location for each feature, thereby providing invariance to these transformations.

4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

For an impression how the sift descriptors look like, see figure 1. Their positions are illustrated in the image with arrows.

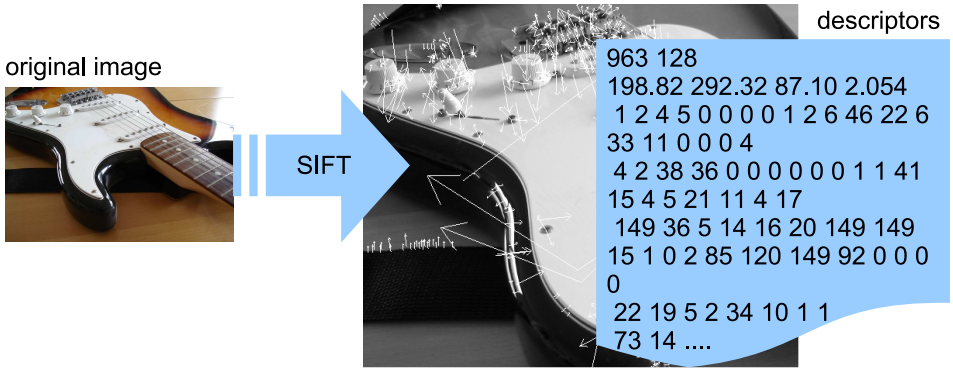


Figure 1: Transformation from an image into feature vectors.

To check for a match, the thereby received keypoint descriptors are compared with those of a reference image. The more similar descriptors, the higher the match. This way one can find the best matching picture within a database.

2.2 Approach 1: Server Side Calculation

The mobileor framework provides the possibility to do the calculation - the SIFT and the matching - on a server. This is because mobile phone processors are very slow compared to a normal computer. Also memory resources are very limited on a mobile phone. Therefore, as a first step the picture taken with the mobile phone has to be sent to the server. After the server has finished its work, the result (depending on the choice of the application. See 1.2.1) has to be sent back to the mobile phone. So let us now have a look at the communication.

2.2.1 Communication with Server

The in 1.1 described possibilities have been considered. For the first part, where the image is sent from the mobile phone to the server, a method for file transmission is needed. Bluetooth is - due to the easy handling and because it is free - an option.

But the limited connection range would be a significant disadvantage for the mobileor framework. A user would be bound to a bluetooth sender and therefore no real mobility would be possible. For object recognition on mobile phones such near distance communication is not appropriate. A user should not have to be always close to a server. For the same reasons infrared and other ways of near distance communication are also discarded here. Bluetooth is just used to upload reference images (see 2.2.2) in this work.

There are only a few far distance data transfer possibilities on a mobile phone. All of them are not free of costs. One is SMS which is due to the text limitation not usable for images. Another is the Internet access (either through WAP, GPRS, UMTS or similar services). But here the applications differ with device, manufacturer, and provider. Available technologies range from the poor WAP up to complete UMTS browser applications specially designed for mobile phones. So an upload possibility working for all the different mobile phones and providers is not possible without e.g. a Java application on the mobile phone which would give back the compatibility. Mobileor provides an application for doing this (see 2.2.3), but it should be possible to use object recognition without having to install something on your mobile phone. Usability is an important issue, especially for commercials. Therefore communication over the Internet access on mobile phones was discarded for this project.

Especially for this semester thesis, where also images and not only text should be sent, MMS gives an easy and widely used messaging possibility. So the user does not have to install anything before the service can be used. Most of the mobile phones which are able to photograph have a built-in option to send a picture right after it was taken. Also the standardization of MMS comes as an advantage for mobileor. MMS consists (like emails) of several parts which can be either multimedia or a text elements. Mobile phones and system providers offer the possibility to send an MMS to an email address instead of a phone number. On the target mailbox just a normal email will arrive which contains the image. The mobileor server can easily load the images from an IMAP service of the mailbox. Therefore, in this work MMS was chosen as the transport medium from the mobile phone to the server.

For the return path MMS could also be considered. But for this project, SMS was chosen to reduce costs. The first application should only return some text, like a result or an URL. So no multimedia possibility was needed so far. Of course, for future versions they should also be considered.

2.2.2 Processes on the Server

Given the communication methods, an appropriate server side implementation can be established. The following steps have to be done (cf. figure 2):

1. **collect messages**: The first step is to take the emails out of the mailbox and read

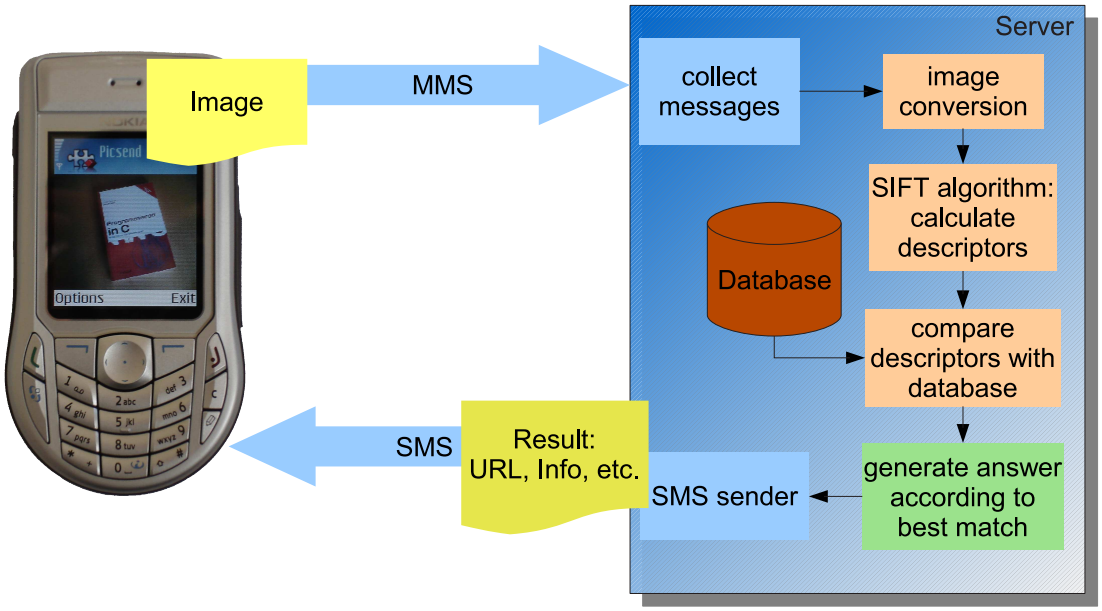


Figure 2: Overview of the setup with server side calculation.

it. Here the image is separated from the text and also the sender has to be stored for to send a response later.

2. **image conversion**: To use the image as input for the SIFT, it has to be converted into a grayscale image.
3. **SIFT algorithm**: Calculation of the descriptors.
4. **matching**: The query descriptors are compared with the descriptors of reference images from a database. The result is the reference image which matches best to the input.
5. **generate answer**: According to the result an answer is generated for the user. This can be information about the product, an URL or simply the name of the best matching reference image.
6. **SMS sender**: Eventually the answer is sent back to the mobile phone by SMS.

As database a collection of reference images together with their descriptors is kept on the server. They can be photos taken with a mobile phone camera which are uploaded by bluetooth or also from other digital cameras or sources. To feed the database correctly with new reference images, a shell script was implemented. It takes as input a folder containing only jpg images. After copying them into the reference folder, the script converts them into the portable greymap file format (PGM). Then it applies the

algorithm to extract the descriptors and saves them into a keypoint file located in the reference folder too.

2.2.3 Symbian Application

To simplify the sending process on the mobile phone, so that a user does not always have to enter or select the target email address, an application was written for the mobile phone. It consists of the functions to take a picture and send it directly to the server, what gives a better usability for frequent users. It assures also that MMS messages generated with this application are all of the same form. So there is no text before the image, two images within one MMS, or similar things corrupted by typing errors or incorrect handling. This way it would be also possible to ensure that a sent image has always the same size or format.

2.3 Approach 2: Calculation Fully on Mobile Phone

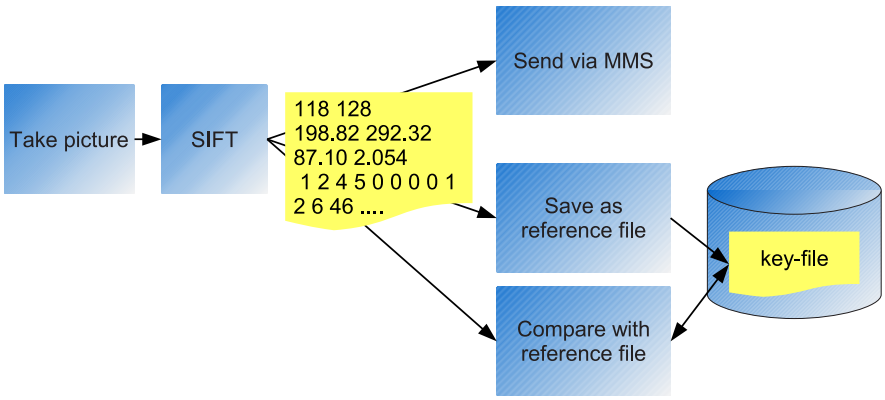


Figure 3: Overview of the setup with the calculation completely on the mobile phone.

As a second approach of this semester thesis, the object recognition was completely ported to the mobile phone. This resulted in a completely independent application which does not need any server or connection. Since the resources of a mobile phone are very limited, the most challenging part of this approach was the implementation, which is described in 3.2.

An overview of the processes running on the mobile phone can be seen in figure 3. As in the server based approach, after the image is taken, the SIFT descriptors are calculated. The application then offers the possibility to save these descriptors on the permanent store of the mobile phone. So later on you can take a new picture, calculate the descriptors of it and compare them with the stored ones. If it matches sufficiently well, the application will report it on the phone display. The algorithm for the matching

is the same used by Lowe. In addition to match with a saved keypoint file, the keypoint file can also be sent by MMS to an email address. This was used to compare the descriptors with those calculated on a normal desktop computer, and to verify their correctness.

3 Implementation

In this section the implementation of the two approaches is described in detail. Also challenges encountered during the development are discussed here.

3.1 Approach 1

Since the implementation of the server side approach is structured into exchangeable parts, they are explained here separately each in one paragraph. For an overview see figure 2 where the colors correspond to programming languages: blue is for PHP, orange for C/C++, and green for shell script. The process coordination and evocation is done in a bash shell script which basically does a loop over the following subsequent processes. The data transfer between the processes is done with queues which are no more than just directories in the file system. At the beginning of each pass of the loop, these queues are emptied.

3.1.1 Message Receiver

The first process is a PHP script that collects MMS messages received by email. It is evoked by the shell script in the beginning of the loop. To handle the IMAP connection and to decompose the messages into text, sender, and picture, a class called MMSService was written. It offers several functions to connect to and disconnect from a mailbox, as well as to list messages, download and delete them. Problems encountered here are the differences between the emails generated by different mobile phone network providers. They have not the same order of text and image and therefore had to be handled separately. MMSService now can be used with all network providers in Switzerland and should be easily adapted to other countries.

The PHP script to receive the messages in mobileor uses this MMSService class. It collects the messages and writes them into files. So for each message there will be files picX.jpg for the image, textX.txt for the message text, and fromX.txt for the sender, where X is the number of the message. These files are written into queue number 1, so into a folder queue1.

The shell script now checks if there are messages in queue 1. If so, it continues with the processes described in the following paragraphs. If there are no new messages, it will wait for some seconds and then call again the PHP script.

3.1.2 Image Conversion

The next process takes all jpg files and converts them by using the convert program of ImageMagick [IMM]. The conversion is needed to get an image in the portable greymap

file format (PGM) which is used by the SIFT implementation of Lowe in the next step. The pgm files are also saved into queue 1.

3.1.3 SIFT

As a third step the feature detection and transformation is applied for each image. The implementation used is the original SIFT implementation in C by Lowe. It takes as input the pgm file from the last step and returns a key file which is a text file containing the resulting descriptors. This key file is saved into queue 2. Also the text and the sender information is now copied into queue 2 for the next step.

To gain a little bit more speed, the image representation of the descriptors was removed from the original implementation. Mobileor uses only the vector form of the descriptors.

3.1.4 Matching

The database with reference images is simply a directory containing pgm image files and their corresponding SIFT descriptor files. The matching of the in queue 2 saved images is now done by comparing the descriptors of each of them with each of the descriptor files of the database folder. For the matching itself, the approach and code of Lowe [Low04] was used, which takes as input two images with their two descriptor files (.key). To speed up the process the part where the visual matching is drawed was removed. The modified C program now just returns the number of matched keypoints which then can be compared with others. The best matching reference image is chosen as the one with the highest number of similar keypoints. For each image in queue 2 the name of the best matching reference file and the number of similar descriptors is written into picXmatch.txt in queue 2.

3.1.5 Answer Generation

Now the answer for the user has to be generated. This is done so far by just taking the pixXmatch.txt file from the last step. So the answer on the mobile phone will be just the name of the matched reference picture as well as how many similar descriptors it has. Of course this can be extended in future versions. For example, a file for each reference picture containing the answer could be stored in the reference folder. It is also possible to prepare a multimedia file (e.g. an image) which will then be sent as MMS.

The for each picture in queue 2 generated answers are stored into queue 3. Also the images itself as well as the sender information are copied into queue 3.

3.1.6 Message Sender

As a final step, the answer has to be sent back to the users mobile phone. So far this is done by SMS, since it will contain only a short text as described in the last paragraph.

Switch [SWI] offers a service for sending SMS. The text is simply sent to a special email address at Switch and will be delivered as SMS to the destination mobile phone. Since this service is free of cost for the ETH, it was used to send the answer in mobileor. A PHP script parses the queue 3 folder for messages and sends each of them to the in the first step extracted mobile phone number.

This last step concludes the cycle of message processing. The shell script empties all queues and starts a new cycle.

3.1.7 Application on the Mobile Phone

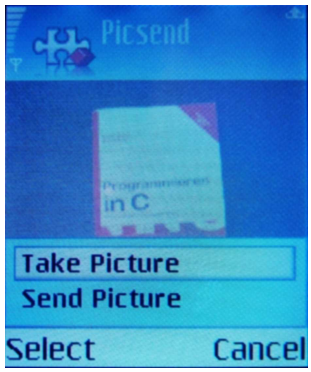


Figure 4: A screenshot of the Symbian application which is able to take and send pictures.

As described in 2.2.3, a small application on the mobile phone was implemented, which offers a function to take a picture and another one to send it by MMS. It was written in Symbian v8.0 using the Nokia SDK. For details about the SDK offered by Nokia, see Appendix A. Symbian offers built-in functions to construct MMS messages and append text or files to it. Also the functions to access the built-in camera are given. Therefore the implementation of this small application did not cause any larger problems and it should also run on all mobile phones with Symbian version 8.0 or higher. In figure 4 you can see a screenshot of the application.

3.2 Approach 2

The second approach was the complete porting of Lowe's object recognition algorithm to Symbian v.8.0. Implemented was one single application which has all the in figure 3 illustrated steps to recognize an object. The user can execute the steps by using

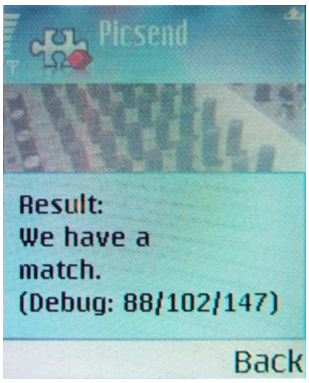


Figure 5: A screenshot of the server-independent mobileor application where it correctly matches two images. The reference image was taken with the larger image size and the actual picture with the smaller image size. In the debug line you can see the number of matching descriptors as well as the number of descriptors of the actual and of the reference image.

a menu. The first menu option, taking a picture, uses the same implementation as the in 3.1.7 described application. The other steps are explained in the following paragraphs. A screenshot can be seen in figure 5

3.2.1 SIFT extraction

This was the most challenging part of the implementation. Mobile phone processors are not as fast as normal desktop computers and the offered memory resources are very limited. It is not possible to use larger arrays on the stack because its size is too small. So to hold images in the memory, heap was used instead.

The heap on the used Nokia 6630 has a size of around 5 MB. This is not enough to hold for example four images of size 640*480 with float grayscale pixels. If you need also other variables or use objects, the free heap size decreases once more. But the SIFT algorithm uses more than only one image in memory. The by Symbian offered camera functions returned images have e.g. on a Nokia 6630 a size of 640*480. To display the image, the original has to be kept in memory separately, since it should be not affected by the SIFT calculation. So the remaining memory size will not be enough to do the algorithm. Therefore it was decided to first scale images to a size of 320*240. Using this size, there is enough memory to perform the SIFT transform.

Lowe uses a custom memory allocation that works with pools. This is useful to avoid too many allocate/deallocate commands. But it uses more than the really needed memory space and had therefore be replaced by specific allocation for every needed byte of heap memory. Also there were some speed up techniques which use larger memory space to become faster in calculation. Some of them had to be removed.

Speed is an important issue on a mobile phone. The algorithm works with 320*240 pixels but is slow. It takes around 50 seconds to extract the descriptors of an image. Therefore two options have been implemented. One works on 320*240 pixels and the second one uses 160*120 pixels. The latter one takes only around 13 seconds and is therefore really applicable for usage. Of course the number of descriptors decreases with smaller image size, but the result - as shown in the next section - is still applicable.

3.2.2 Save Keypoints

After the descriptors have been calculated, they can be stored in a reference file. This file will be on the mobile phone file system and can be used by the application later on for comparing with another image. The stored file will be called refkeys.txt so that it is also readable by the file manager of the mobile phone.

3.2.3 Compare Keypoints

If a user takes another picture and executes the keypoint calculation once more, the new descriptors can be compared with the first set. The application just takes the descriptors in memory and those of the file refkeys.txt and compares them by using a ported version of the matching algorithm by Lowe. Since there is not much memory needed to do this, no problems were encountered during implementation of this part. Also it is fast because with the above mentioned low image sizes, there are not many descriptors.

The result of the comparison is the number of similar descriptors. The application returns a match if they are sufficiently many (relative to the total number of the actual and the reference image). The result is shown on the display in a little message box (see figure 5).

3.2.4 Send Keypoints

To compare the descriptors calculated on the mobile phone, the application offers a function to send them to the server. It uses the same way as for the server based approach: MMS. The keypoint file is attached as a text file to an MMS message and then sent to a given email address. Keypoints transferred in this way have been used for the evaluation in the next section.

4 Results

This section discusses the results of both approaches. To give an impression, also some examples are shown here.

4.1 Approach 1

The server side approach has been tested with a Nokia 6630 and a Linux computer at ETH. The reference pictures have been made with a 5 megapixel digital camera from which a small test database of 30 different pictures of books and some other objects was prepared on the server. The extracted number of descriptors per image is on average 2000.

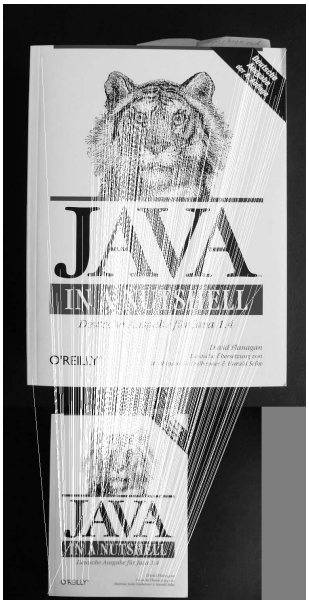


Figure 6: An example for the server side matching with 505 matching keypoints.

The system was started and several test images were taken with the mobile phone and sent by MMS to the server. On average 1000 descriptors could be extracted per received MMS picture. This resulted in a very high matching quality. An example is given in figure 6, where 505 descriptors have been matched.

If the object is photographed without occlusions, then it is always matched to the correct reference image of the test database. Also the viewpoint can be changed a little bit, with still correct results. In figure 7 you can see an example with occlusion. Even though parts of the book are covered by the hand, the result of the matching are still 268 similar descriptors. Since these are very distinctive, such a high number is enough to

match the image correctly with its reference image in the test database.

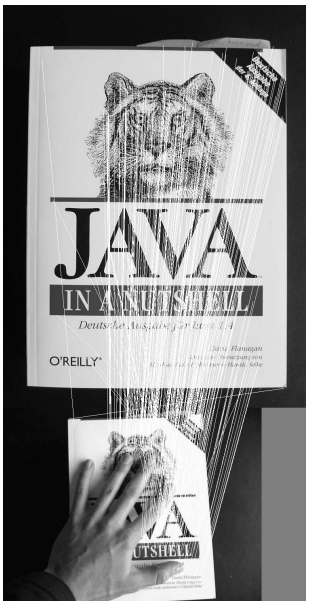


Figure 7: An example for the server side matching with occlusion where still 268 keypoints match.

The only problem with the server based calculation is the amount of time needed for the matching. Even with a small database it takes several minutes which is not appropriate for a user application. This will be discussed in 5.2.

4.2 Approach 2

As mentioned before, the time needed to extract the descriptors of an image was measured on a Nokia 6630. It takes on average 50 seconds for 320*240 pixels and 13 seconds for 160*120 pixels respective. The resulting number of keypoints is low due to the small image size. With the larger image size this is on average 80 descriptors and with the smaller image size 40 respective. The effective number varies with the image details.

To verify the descriptors calculated on the mobile phone, they have been sent to the server where they have been compared with the original image of the mobile phone. To test extreme situations, also the picture from figure 7 was used on the mobile phone. The descriptors of the occluded book have been calculated on the mobile phone. Then these descriptors have been compared with those of the reference image calculated on the server. The results can be seen in figure 8. The example on the left was done with the larger image size on the mobile phone and gives 27 matching descriptors. The one on the right was done with the smaller image size and results in 25 matching descriptors.

Figure 8: These are two examples where the descriptors of the two smaller images at the bottom have been calculated on a mobile phone.

The independent application on the mobile phone does not have a database. Only one reference image can be saved and used for later comparing. In figure 5 you can see the result of an example matching where the reference keypoints have been calculated on the larger image size and the actual (compared) descriptors on the smaller image size.

Since the mobile phone gives only a few descriptors, the success rate of the matchings is not as high as in the approach with the server side calculation. Especially small details are not detected. So to have a good result, the object should be large on the image.

5 Conclusions

This section will give some final conclusions and a brief overview about possible future work in this topic and on the framework mobileor.

5.1 Conclusions

This semester thesis has shown that object recognition can be combined successfully with mobile phone technology. In two approaches, one which uses server side calculation an one which runs completely on a mobile phone, pictures taken with a mobile phone camera have been used as input to object recognition algorithms.

The developed framework mobileor can be used as a basis for several applications of object recognition. As an example a city guide is mentioned which does not need any bar-codes or other tags to recognize famous buildings an places. The framework can be easily extended or parts of it can be exchanged - e.g. the object recognition algorithm -, to adapt it for specific applications.

The results in correctness are promising - even those with descriptors calculated on a mobile phone. Even though there the images have to be first converted in a small and for the memory resources of a mobile phone handable size, the application is able recognize objects. It can be taken as a proof of concept, that object recognition can be implemented fully on a mobile phone.

Especially the results with the server side calculation convince with a very high correctness in recognition of objects. This motivates to develop applications based on mobileor.

5.2 Future Work

Since the speed of the matching process on the server side approach is a great bottleneck, it would be very useful to come along with better solutions for large databases. Special techniques based on hashing or presorting of the descriptors could be used here.

As mentioned before, the mobileor is just a basis. Therefore the implementation of specific applications should follow this semester thesis. To allow also multimedia answers, one could also add the possibility to use MMS instead of SMS for the sending of the answer back to the mobile phone.

The SIFT implementation of the server independent approach, which was basically a porting of the original SIFT implementation to Symbian, could be improved. There are still a lot of possibilities to speed up the process and maybe to realize an implementation which works also with larger image sizes. Ideas of using the file system have been

considered but discarded because it has too slow access times.

On the server side implementation as well as on the mobile phone based approach, it should be tested with other algorithms for object recognition.

A Symbian HowTo

This Appendix describes how to install and use the Nokia SDK for Symbian.

A.1 Disclaimer

In this document I describe my own installation steps. I do not guarantee that this will also work on your computer. Therefore, I assume no liability for any damage caused by following this installation steps.

A.2 Windows

This section describes first how to install all needed tools. In the second part, a little example is compiled and started.

A.2.1 Installation

Symbian SDK

Nokia offers an SDK called ’Nokia Series 60 SDK’:

- Go to http://www.symbian.com/developer/sdks_s602EcppFP2.asp
  Depending on your mobile phone (version of Symbian), you have to choose the right version. For the Nokia 6630 used for the semester thesis I used the ’Series 60 2nd Edition with Feature Pack 2 (Symbian OS v8.0)’ because the 6630 has Symbian OS v8.0.
- Install it. It is important to keep the default installation path, since there is a bug.
- Go to http://www.activestate.com/Products/Download/Download.plex?id=ActivePerl and download the latest MSI-File and execute it to install ActivePerl.
- Add the following paths to your PATH environment variable (In the Startmenu: ’System manager’ - ’System’ - ’Advanced’):

```
C:\Program Files\Microsoft eMbedded C++ 4.0\EVC\wce400\bin
C:\Program Files\Microsoft eMbedded C++ 4.0\Common\EVC\Bin
```

Eclipse

If you don not have eclipse, install it first:

- Go to http://www.eclipse.org and click on ’download’.
- Download the newest version and follow the installation steps.

Now install CDT, which is the C/C++ Plugin for Eclipse:

- Start Eclipse and click on ’Help’ - ’Software updates’ - ’Find and install’.

- Select ’Search for new features to install’ and click on ’next’
- Click on ’New Remote Site’. Now you are asked to enter an URL. Here is how you get the correct URL:
  - Open a Browser and go to http://www.eclipse.org/cdt/
  - Click on ’Downloads’ (listed under ’General Ressources’ and NOT on the menu on the left).
  - Under ’Current Releases’ you will see a link titled with ’Use the following URL in a Site Bookmark in the update manager:’.
  - Copy this link location and paste it to Eclipse.
- Back in Eclipse, click on ’finish’.
- Toggle all search results to be installed and click on ’next’.
- Accept the license (if you want ;)) and click on ’next’.
- Now click on ’finish’. CDT will be installed now.
- If you are asked to restart Eclipse, do it.

A.2.2 Example: Hello World

Mostly Manually
The Symbian SDK comes with several examples. Here it is explained how to import an example into eclipse, compile it and eventually run it in the emulator.

- It is useful to create a folder for your eclipse projects. I use

```
C:\Symbian\8.0a\S60_2nd_FP2\epoc32\BUILD\SYMBIAN\8.0A\S60_2ND_FP2\WORK
```

- Create a new ’Standard Make C++ Project’, since the makefile has to be written by hand.
- In the Window ’C/C++ Projects’, right-click on the new created project and choose ’Import..’.
- In the import wizard, select ’file system’ as import source.
- Select the directory containing the helloworld example.
- Create a file called ’makefile’ with the following content:

```
all:    ABLD.BAT HelloWorld.h HelloWorld_Application.cpp HelloWorld_AppUi.cpp HelloWorld_AppView.cpp HelloWorld_Document.cpp
      HelloWorld_Main.cpp Bld.inf HelloWorld.hrh HelloWorld.mmp HelloWorld.rss
        abld build wins udeb

clean:
        del ABLD.BAT
        del /Q C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\wins\udeb\z\system\APPS\HELLOWORLD
        bldmake clean

ABLD.BAT:
        bldmake bldfiles
```

- Now you can create the make targets ’all’ and ’clean’ in eclipse (right click on the project).
- Build the project. (clean and then build). If there are one or two errors reported, starting with ’EDLL.LIB *...’, just don’t care about that.
- Start the ’Emulator (debug)’ from within the Windows startmenu.
- On the mobile phone emulator there should be now a new application in the main menu called ’helloworld’. Run it.

To run it on your real mobile phone you need to first create a sis-file (this is something similar to the msi-files of Windows):

- Create a file called ’helloworld.pkg’ with the following content:

```
; HelloWorld.pkg
;
;Language - standard language definitions
&EN

; standard SIS file header
#{"HelloWorld"},(0x10005B91),1,0,0

;Supports Series 60 v 0.9
(0x101F6F88), 0, 0, 0, {"Series60ProductID"}

;
"C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\armi\urel\HelloWorld.app"-"C:\system\apps\HelloWorld\HelloWorld.app"
"C:\Symbian\8.0a\S60_2nd_FP2\epoc32\data\Z\system\APPS\HELLOWORLD\HelloWorld.rsc"-"C:\system\apps\HelloWorld\HelloWorld.rsc"
```

  Of course, you have to adjust the ID number to yours.

- Open a Command Prompt.
- Change to the directory where your source files are (the eclipse project directory).
- bldmake -v bldfiles
- abld build wins udeb ⇒ build it for the emulator
- abld build armi urel ⇒ build it for the mobile phone
- The files for the mobile phone are now in C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\armi\urel
- makesis helloworld.pkg
- Now there should be a helloworld.sis in this directory. This is what you have to send to your mobile phone. E.g. you can use bluetooth:
  - Activate bluetooth on your computer and on your mobile phone.
  - Open a Windows Explorer and go to the folder where your .sys-file is.
  - Right-click on the file and select ’Send to’ - ’bluetooth device’.
  - Select your mobile phone and send.
  - The mobile phone should detect the file and automatically asks you to install it.

Everything in Makefile
This is a faster way to do it. It does all the build and packaging within the makefile.

- Change the 'makefile' to:

```
all:    ABLD.BAT HelloWorld.h HelloWorld_Application.cpp HelloWorld_AppUi.cpp HelloWorld_AppView.cpp HelloWorld_Document.cpp
      HelloWorld_Main.cpp Bld.inf HelloWorld.hrh HelloWorld.mmp HelloWorld.rss
        abld build wins udeb
        abld build armi urel
        makesis helloworld.pkg

clean:
        del ABLD.BAT
        del helloworld.SIS
        del /Q C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\wins\udeb\z\system\APPS\HELLOWORLD
        del /Q C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\armi\urel\HELLOWORLD.*
   del /Q C:\Symbian\8.0a\S60_2nd_FP2\epoc32\BUILD\SYMBIAN\8.0A\S60_2ND_FP2\WORK\HELLOWORLD
        bldmake clean    bldmake clean

ABLD.BAT:
        bldmake bldfiles
```

- Build the project in Eclipse. (clean and then build)
- Now everything is ready. The application should be in the emulator. Also the sis-file is ready to send to your mobile phone.

A.2.3 Typical Errors

IMPORTANT: Some errors are not correctly shown in Eclipse. So you have to sometimes call make in a console to really see the whole output.
In addition to this, some errors are not easy to understand. Here are some hints:

Static Variables
If you get an Error:

```
ERROR: Dll 'YOURAPP[10005B89].APP' has initialised data.
```

then you have declared global writable variables, which is not allowed in Symbian, since it produces DLL's for running in ROM. More information about this is in the Symbian v8.0 Documentation in the paragraph 'Writable static data in DLLs'.

Small Stack
If you get an Error:

```
error LNK2001: unresolved external symbol __chkstk
```

Then you have put too much in the stack. Mostly this is because of large arrays. To avoid this problem, use the heap instead of the stack.

Problems with floating points in fprintf
If you are using fprintf from the c standard library with floats, then make sure that you change the language of your mobile phone to English. With German, Spanish, or other European languages floats are written with a comma instead of a dot. This can cause errors if you write floats into a file and parse it later on.

A.3 Unix/Linux

There is a method to run everything also on Unix/Linux, but it uses Wine for the emulator (Imagine running an emulator on an emulator). I have not tried it yet. Here it is:
http://gnupoc.sourceforge.net/

A.4 Links

Links:
http://www.eclipse.org
http://www.symbian.com

References

[IMM] *ImageMagick, www.imagemagick.org*.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *ICCV*, pages 128–142, 2001.

[MS03] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Computer Vision and Pattern Recognition*, 2:257–263, June 2003.

[NZZ06] *Neue Zurcher Zeitung, NZZ Online, www.nzz.ch*, 2006.

[Roh05] Michael Rohs. Marker-based interaction techniques for camera-phones. *2nd Workshop on Multi-User and Ubiquitous User Interfaces at IUI*, January 2005.

[RW05] Alexis Rondeau and Stan Wiechers. Semapedia - the physical wikipedia. September 2005.

[SM97] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–534, 1997.

[SWI] *Swiss Education and Research Network, www.switch.ch*.

[SZ02] F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. *Challenge of Image and Video Retrieval*, pages 186–197, 2002.

[TA206] *Tages Anzeiger, www.tagesanzeiger.ch*, 2006.