Philadelphia Image Mobile Search and Recognition

Class of 2009 Senior Design Project
http://www.seas.upenn.edu/~knichel/pisr/index.html
Matthew Evans - evansmf@wharton.upenn.edu
Mark Knichel - knichel@seas.upenn.edu
Faculty Advisor: Jianbo Shi (primary) and Kostas Daniilidis
Abstract

We have created a mobile device image recognition platform, backed by a “cloud-computing” server architecture. The image candidate set consists of buildings on the University of Pennsylvania campus. When a search query is performed with an image captured on a mobile device, the service returns the name of the building.

The software consists of a mobile platform client (Google Android or MMS messages) and protocol, an image indexing server platform (hosted on a cloud environment), and a web gateway server, processing requests and delivering results.

Introduction

Image-driven search remains to be conquered by the search giants of the web. Web search engines crawl the text-heavy web, but our world is a visual place with definite visual features. Very little has been done with genuine computer vision-driven image search.

Visual navigation most clearly follows the way humans navigate. We look for unique landmarks, recognizable patterns, or distinct colors in our attempts to navigate unfamiliar terrain. GPS technology provides macro-navigation, but we rely on visual cues for low-level identification.

Most current computer vision research focuses on face detection, single feature recognition, or motion techniques. Buildings are a difficult task, loaded with unique potential identification features.

A persistent web service, with accurate computer vision algorithms, accessible from mobile devices, can offer an effective visual navigation tool. As we build such a system, the questions become: How can we uniquely specify the visual fingerprint of a building? What image features are most accurate for comparisons? Philadelphia Image Mobile Search and Recognition probes these questions in an exploration of our system.

Related work

Computer Vision Theory

In the theoretical realm, research has been advancing on visual vocabulary trees (Kumar), scale invariant feature transform, or SIFT (Lowe), and color histograms (Shim). There has been little formal research on building recognition specifically, although there has been much research on other types of object recognition such as face recognition. OpenCV represents an open-source package combining the state-of-the-art computer vision algorithms into a comprehensive suite (OpenCV), and how-to books have been published (Bradski). The strengths and weaknesses of these algorithms are detailed in the Technical Approach.

Photo recognition software has been attempted by researchers at the University of Cambridge (Randerson), but progress appears to have stalled in 2004. There have been other attempts as well, but none have become very successful.
A similar topic was tackled in Spring 2007 as a Senior Design Project (Aleong). In that case, users captured photos offline, downloaded them to a PC, and emailed them to the project leader, who then used a laptop with Matlab and Outlook 2007 to produce results. Our project aims to provide a sustainable service with more automated server architecture. Aleong sought to build a mobile, automated tour guide application.

**Mobile Image Search Applications**

Recently, an application called SnapTell was released for the iPhone that allows you to take a picture of the face of a product and it will return results that compare prices of that product on different websites. SnapTell currently runs only on the iPhone platform, and only works on products - it is designed for comparison shopping. Early reviews claim it performs very well (Kincaid). SnapTell is apparently also building a service via MMS to detect book, movie, or game covers.

The service like.com is a website that allows users to search for product images and then find other products similar to that image in various dimensions. In using it, a category is selected (shoes, handbags, etc). Like.com will then use image similarity algorithms to find other products that are similar to a selected product on a given dimension. Our project is different because we return keywords and data based on images in general that the user supplies. Like.com tunes its algorithms specifically to the products in question and is not applicable to location or building pictures.

The service Tiltomo (at http://www.tiltomo.com/) allows users to view images and then search for similar images based on Theme or Color / Texture. Those are the only two dimensions users are able to search over. Tiltomo indexes all its images from public images on the photo sharing site Flickr. Also, users are limited to the pictures that Tiltomo has indexed off Flickr. They cannot provide their own images to find similar images. Our service is different because we will not allow users to search for other images based on similarity. We return keywords based off what image a user provides. Although at first we are limiting the types of pictures that users can submit to buildings, we will seek to be more general in the similarity matching process.

GazoPa (GazoPa.com) is a new image search engine (still in closed beta testing). Searches can be made with normal keywords or search based on color and shape. Once you have found an image, you can then find similar images. Also, you can upload your own image and search for images that are similar to it. Last, you can draw a sketch of what you are searching for and find images similar to that. GazoPa has indexed over 50 million images.

Retrievr (http://labs.systemone.at/retrievr/) is very similar to GazoPa but with less functionality. You can search by sketch and image, but not by keyword or description (color / shape) of the image. Our product is different because when a user submits a picture, we will be able to return keywords and information associated with that image instead of just similar images.

Recently, Microsoft and Amazon have released products that try to compare images. However, in Microsoft’s and Amazon’s case, these products actually do not use computer vision to achieve their results. Amazon uses a service called Mecahnical Turk which utilizes humans to service their iPhone application that can use a picture to search Amazon’s large product database. Microsoft has a feature in their image search to search for similar
images; however, this uses meta-data and not image features. Although this is a step forward in the image recognition field, we are disappointed in the lack of computer vision used in these products.

Several desktop software suites claim they can find similar images on a machine and remove duplicates. Most of this software is not general enough and does not find images that are not almost exact duplicates. Additionally they only remove images, can only work on images on a single machine, and do not provide any information about the images.

**Message Delivery Applications**

On the text/multimedia message delivery front, there are several services that provide automatic SMS-sending interfaces (Fehrenbacher) although not all have proven sustainable (Malik). Teleflip offered a convenient mapping of anyphonenumber@teleflip.com allowing SMS over email. Peekamo, txtDrop and GizmoSMS offer free web-based text message submission, or in some cases direct API licensing, along with advertising, in an attempt to create advertising delivery vehicles. These services merely illustrate the ease of email-text integration - they do not provide any content or application functionality.

None of these similar services provides the same functionality that our project provides. While our project shares similar techniques to search for similar images as these services, we return a different end result. We hope to be able to search over as many images as GazoPa does very efficiently, and provide more useful information than just returning similar images. We believe that the ability for users to receive a keyword, website, and other web resources given a picture is very valuable to the mobile market and has not received enough work and research. We feel that combining the mobile focus of a product like SnapTell and the image-driven search of an engine like GazoPa into a location-based product provide a clear benefit for mobile phone users in the location-based search space. We only use features extracted from the images and no meta-data to achieve our results.

**Technical Approach**

The project consists of challenges in three major areas. First, mobile interface design requires learning APIs and programming with limited resources. To limit the challenge in this area, we developed on the Multimedia Messaging Service (MMS) protocol first, a form of advanced text message that interfaces well with email. Second, constructing the image recognition engine required learning computer vision concepts and applying machine-learning theories. Third, the amount of processing needed and the parallelizable nature of computer vision tasks suggests the use of cloud-based computing and encourages us to exploit those resources. Amazon.com offers a cloud-computing platform, including flexible storage, scalable processing, and low-cost availability.

The end result allows users to take a picture of a building from their mobile phone and send an MMS message or use an Android application to query our server. Our server processes the image, compares it to all the images in our database, and returns the name of the building in the image. In its current incarnation, there is much more data available for buildings near the Engineering quadrangle, and as a consequence, results are better for those buildings.

Users are able to take a picture on their mobile phones and use an Android application or
send a MMS message to our service with that picture attached. When a message is received, the picture is extracted, and the vision algorithms are run to find the most similar images. The building name is returned to the user by sending them the keyword in another SMS message or by returning the value to the Android application. SMS messages are received via an email account. SMS messages are sent back to the user by sending an email to a specific email address, depending on the cellular carrier of the user. The mobile application allowed us to create a much richer interface for the user. However, since Android is not a widely used operating system, we still support the MMS system.

Building the database

First, to create a functional system, a database was built of images taken manually of the campus to use in our recognition process. These images were taken at many different angles and in different weather and lighting conditions to ensure good candidate images to match against. Initially, 550 images were taken of 31 Penn buildings surrounding the Engineering quadangle, and in a second dataset, 3500 more pictures were acquired of the rest of campus. Additionally, using OpenCV, perspective and scale transforms are performed on these images and the results are stored. This increases the accuracy of the system because it deals with potential tiny perspective and scale differences between the dataset images and the test image. The test dataset was drawn from two sources: images downloaded off the Internet from Google Image Search and pictures taken of the buildings to include only in the test dataset.

Image features

Four features from the images are used in the system: HSV color histogram, mean pixel values, corners and edges although a number of other options were explored. Colors are one of the most obvious features of a building. It is very easy to calculate color statistics of images and can be easily compared. Edges and corners are also usually very distinct in images of buildings as well.

The pictures are converted to the Hue-Saturation-Value (HSV) color space before calculating a color histogram. This color space is supposed to remove some of the variance due to lighting and shading. This is very important in images of buildings because the time of day and weather conditions significantly affect the colors in the image. Additionally, before the image is analyzed, the system attempts to remove the sky from the image. Since a lot of images of buildings include the sky, it has a large impact on the color statistics of the image. A naïve algorithm is used that analyzes the pixel values of the image in the HSV space. The color of the sky only represents a small range in the color space. Additionally, this range of values often does not occur in the pixels of the buildings on campus, although this approach could be a problem in other locations. Therefore, any pixel in the image inside the range is converted to pure black. This approach works surprisingly well only removing pixels in the sky and no other pixels in the building.

The histogram is calculated using a built-in OpenCV function. The histogram only looks at the hue and saturation components of the image because value is supposed to represent the amount of light that the image is receiving. Therefore, it would negatively affect the image comparisons. There are a number of parameters for the histogram including the number of bins and the range of pixel values that should be analyzed. The system currently looks at 12 bins for each component and excludes true black pixels which are used for the sky. Then the histogram is converted to a vector to be stored in the database. The histogram is laid out row by row in a vector. However, a transform using a Fast Fourier Transform might convert the histogram into a better vector space that would enhance
comparisons. This was not implemented in our system.

The color histogram represents a global feature. However, the system also looks at local color features. Local color features can help distinguish between two buildings that have similar colors but different shapes. We hope to counter the high dependence on very similar camera angles for local features with a larger number of database images. Local color features are calculated by converting the image into a twelve by twelve grid. Again, different grid sizes were tried and it was found that twelve represented a balance between how local the color features were and the problem of being too specific. The mean pixel values of each grid are calculated, and these values are stored in a vector for comparison.

Next, the image is passed to OpenCV’s corner detection function. This function looks for strong features in the image. Some preprocessing is done on the image before it is passed to the function. Since there is a lot of noise in normal images, including trees, leaves, and building features such as bricks and numerous windows, the corner detection function often finds many incorrect corners. However, eroding the image a number of times removes a lot of this noise and therefore the function will produce more relevant corners. The positions of the strongest corners of the image are stored in the database for comparison later.

Lastly, OpenCV’s edge detection algorithm is run. The system performs some preprocessing on the image to improve the quality of the edges detected. The image is first run through the Canny algorithm to convert the image to a series of line segments. No line data can be extracted from this though. Therefore, the system also tries to remove small line segments and lines that do not appear to be edges of a building. This is done by using a moving window that looks in the neighborhood of the line segment to determine what else is near the line. The edges of a building should be reasonably isolated in these images. Although this greatly increases the computing time of calculating features per image, it produces better results. After this is done, then the image is passed to OpenCV’s edge detection function. The x and y position, the length of the line, and the angle of the line relative to the x-axis are stored.

A number of other features were looked at but were not very successful in detecting and comparing buildings. For instance, Scale-Invariant-Feature-Transform (SIFT) and Speeded Up Robust Features (SURF) are popular algorithms for object recognition. These algorithms aim to find “important” points in the image and store them in a way that makes them scale and angle invariant. Open source implementations are available online and were used to test the algorithms with our images (Bay et al). Visual vocabulary trees made the comparison of these features very efficient, and time was spent researching these structures. Testing showed that these algorithms were not successful at comparing and finding similar buildings. A likely reason is that it is hard for these algorithms to find important points in general building images. This class of images often has a lot of noise including landscape and building features like brick and windows that confuse the algorithm. Also, the algorithms calculate a few hundred features while there are a few thousand images in the database resulting in many false matches.

Many other OpenCV functions were not successful either. Dilating the image, the opposite of eroding the image, did not improve the line detection algorithm. Dilation should make shapes more distinct in the image. However, this increased the number of false positives detected. Additionally, looking at connected components and contours of the image did not work either due to the noise of the image and the different parts and colors of the buildings. Image moments are often used to summarize information about an image. OpenCV implements a number of functions that deal with image moments; these too proved unsuccessful of improving the accuracy of the system.
OpenCV has implemented a match template function that takes a small template image and tries to find a good match in a larger image. If it is possible to find strong points of interest in an image, then it should be possible to take a small area around those points and use template matching against the database of images to find matches. However, due to the large amount of images in our database, there were too many false positives. Taking larger areas around the points decreased the accuracy also due to the fact that the features in the different images are at different angles and colors.

Feature Comparison

After the image features are calculated on the image input, the results are compared to our dataset to find good matches. It is a computationally expensive process to compute the similarity against the database of images. OpenCV's implementation of support vector machines (SVMs) is used to compute the predictions for the color histogram and the mean pixel values. The SVMs are trained on the database offline so that penalty is not incurred every time the service is called. SVMs are a good choice because the data is stored in vectors and traditional vector distance functions are appropriate for that type of data. A linear kernel is used since it proved better than other kernels, such as the generally good radial basis function. The edges and corners are compared using a brute force comparison throughout the database. This does not take too long since the database is relatively small. The four components of the edge data and the positions of the corners are compared and there is a little leeway allowed to account for the slightly different images that are being compared. The number of matches is stored for each image and then the best matches of each building are compared.

Other methods of comparison were also evaluated. Brute force searches tend to be more accurate but also take more time, to the point of impracticality. The SVM method proved to be about accurate as the brute force search for color histograms and mean pixel values. However, SVMs are much faster and OpenCV provides an easy to use implementation. A vector comparison method such as K-Nearest-Neighbor or SVMs was tried on the edge data. However, this proved to not be accurate so the brute force comparison is used. The increase in running time is worth the improvement in accuracy.

Looking at the results of this method, we have been able to achieve about a 35% success rate for recognizing test images. This is vast improvement over the 0% we were achieving with the SIFT method, and improved from our earlier project iterations of 20%, although this percent is still very low. There are a number of parameters that can be shifted that change the recognition process including the number of bins in the color histogram, the range of pixels examined by the histogram, the number of edges used in matching, the construction of the edge and color vectors for comparison, and the leeway for matches. Testing has been done to optimize the parameters but due to the complexity of changing these parameters, the global optimum has likely not been found.
Sample Results

A comparison of two similar images of Irvine auditorium. The corner detection, edge detection, erosion, and color histograms techniques are illustrated (clockwise from top-right).

Transmission/Communication System

As we have explored the MMS and SMS interface environment, we have discovered that email can be used to interact with text (Esengulov) - and picture-messaging systems on many (possibly all) major carriers, domestic and international. These carriers enable users to send text messages to email accounts, and to receive text messages from a special email address (often of the format phonenumber@carriername.com). With the advances by carriers mapping text messages to email, our interface task is narrowed to creating an email "frontend" for our "backend" services of image matching and data lookup.

To build a mail-response server, we used the Java Mail API, part of the Java Enterprise Edition (EE) platform. Java code driving this API is bundled using the Java/Apache Struts framework, a toolset that allows us to connect web pages, Java mail server requests, OpenCV executions, and our image dataset. We have used a GoogleMail address for this project.

Server Architecture

We have divided the servers running our project into 3 separate process servers: a
MySQL Image Database running on its own machine, a mail- and web-server interface server, and an OpenCV image processing image. Due to the nature of the OpenCV execution, it is located on the same physical server as the interface server. OpenCV makes several database calls in its execution to compare image statistics.

Challenges

The most difficult challenges that we have encountered have been finding good image features. We spent a lot of time researching and trying out SIFT and SURF implementations given their good recommendations. However, these yielded no results. Although other methods have been suggested and researched as well, so far it has been difficult implementing these different methods and testing their success. Additionally, many of the suggested algorithms have proved ineffective on building images.

It was also difficult finding the best parameters for calculating the features. It is possible to tweak the parameters of all the methods used. Changing some of these can produce drastically different results. However, it proved difficult to test which parameters were better than others due to the time intensive process of calculating the features for all the images and then running the test set through the program. That would have been prohibitive. Instead, a few images were used and the results were looked at by eye to see which methods produced better results. Although this was not thorough or error-proof, it provided a useful working alternative.

Design
Resources

The project required several key resources. Our laptops were the primary development and test environment, while server resources are required for full-scale demonstration and production use. Storage has been used for image collection, and initial tagging and storage. We used our own cameras, and existing image sets, for image capture.

We are using the Amazon Cloud Computing platform for easily and cheaply scalable work. The simple storage service (S3) platform offers virtually unlimited storage and database options for our image library. The elastic compute cloud (EC2) offers instantly scalable computing resources, and can be deployed by launching an uploaded disk image. We run a Linux distribution on the Amazon platform, utilizing the Struts framework of Java and Apache, running in a Tomcat/Jetty application server instance. We have been fortunate exploring educational discounts with Amazon - they have offered us some credits on the AWS platform, and we estimate the remainder of the service charge to come to less than $10.

We use the cloud for image processing, to launch instances on demand, and to run the comparison algorithms. Since cloud computing is an interest of both of us, we have used the Amazon Web Services tools even though a simpler platform might also suffice.

A web server in the cloud for sending/receiving emails to text messages serves as our initial delivery gateway. While we may later transition to an iPhone service receiver, an email server performs the basic message receiving and sending capabilities we need to implement the project. Open-source email servers are available, and have been integrated into our web service stack for deployment on the Amazon servers.

To demonstrate a mobile device platform, an iPhone or Android would be required. In demonstration, we have used an Android emulator running on a laptop computer. With the SMS/MMS format, we can use any mobile device with a camera phone. Text and SMS messages are a standard which can be easily applied to email frameworks.

Conclusion

Philadelphia Image Search and Recognition is a mobile device image search system that is able to autonomously recognize buildings given a picture from a user. The system only uses visual features to find similar buildings – no metadata is used. Users are limited to buildings on the University of Pennsylvania campus, but the approach to the system is scalable to more locations.

Users take a picture of a building they are interested in with their cell phone. Then, using either MMS or an Android application, they send the image to our service, which runs various algorithms, and returns the name of the building.

Further improvements on this idea could include returning more metadata and useful
information than just the building name. Also, more than just visual features could be used, such as GPS location if available. We did not include that in our project since we wanted to focus on the computer vision aspects of the system. The structure of the system could also be adapted to a parallel system so that a larger database of images could be used over a wider area. This system would also be much faster than our current system. The issue of scale and angle invariance is a large problem in the current system. Some approaches have been made to solve this problem (Hutchings) but more research needs to be done before this set of techniques is mature.

This project was an ambitious project given our limited knowledge in computer vision. We hoped that we could use previous computer science research and open source implementations to achieve our goal. This proved harder than we had anticipated. Unfortunately, there has not been very much research into building recognition or general image comparison. The methods introduced in the few papers that do exist are hard to implement given our limited experience. Additionally, there are not as many open source implementations of these and other computer vision techniques as we would have hoped.

The end result of the project is a blend of multiple technologies that are will become much more prevalent in the future. Our project has shown that such a system is possible. We anticipate that many more companies will soon release products similar to our project.

References


Aleong created an image lookup program for her senior design project at Penn’s CIS department. This is her final paper from the project. This served as a reference for what projects have been accomplished in this context in the past, as well as what resources are available at Penn. Interestingly, this paper was not found until the end of the first semester.


This is the published guide and documentation for the Amazon Web Service platform. Amazon offers computing, storage, database, and queuing services on demand. It is the definitive source for any projects running on the platform, using its EC2 compute cluster, S3 storage, or other resources. This is also a community portal for peer-to-peer support and communication. This site served as our primary reference for AWS information in the building of our server platform.


SURF represents the latest research in SIFT-related algorithms. We consulted this paper and related site material for implementations of the SURF algorithm.

Bradski and Kaehler offer the closest thing to an official manual for the OpenCV library. It is published by the well-known O’Reilly, which is possibly the most authoritative computer science literature publisher. We consulted this book for information as we tweaked OpenCV’s settings.


Esengulov’s blog post on the popular makeuseof.com blog was picked up by popular social networking sites, where it was noticed by our team. While the blog itself and Esengulov can hardly be considered reliable, this page serves as a regularly updated aggregation point for text-messaging domain names for many major carriers. The recent publication date reflects a recent update to the post – the content has been published for a much longer time. We used this information to identify a bridge for email-MMS messages.


GigaOM is a widely-read technology industry blog. This post follows the emergence in the marketplace of "free" text message transfer services, and offers editorial content about their viability as businesses. We used this information to evaluate the feasibility of text/MMS messages as a medium.

Hutchings, Robin, Walterio Mayol-Cuevas, Building recognition for mobile devices: incorporating positional information with visual features. CSTR-06-017, Computer Science, University of Bristol. December 2005

This paper discusses the approach of using GPS coordinates and scale-invariant planar representations of buildings in a navigation system. The system is unavailable for test, but we were able to reference this work in looking for advanced navigation-related techniques.

JavaMail API, Java 6 EE. http://java.sun.com/products/javamail/

This is the published API home page for the JavaMail API in Java EE. JavaMail is a series of tools to interact Java server programs with several email protocols. We were brought to this resource via other work with Java EE, and this site offers tutorials and other reference information for building with the JavaMail API. We used this to learn about a feasible option for building our reception-response email system.


Kincaid’s review on the well-known blog TechCrunch informed us of the recently launched SnapTell service. Kincaid tested the service and found it was quite successful, marking a serious competitor to our idea. SnapTell focuses on retail product recognition, and uses both the iPhone and text messages as platform. Kincaid’s review is trustworthy (and easily repeatable).
Philadelphia Image Mobile Search and Recognition


Kumar’s thesis and related published article combine the work of a visual vocabulary implementation with a time-synchronized image set to produce loops of images from a camera. Her work represents a strong use for vocabulary trees. While it comes from reputable scientists at a reputable university, we were unable to easily reproduce the success with visual vocabulary trees that Kumar found. We used this to understand the complexity in physical building image recognition.


Several computer vision experts (our faculty advisers) and documents pointed us to Lowe, a researcher at the University of British Columbia, as an authority on SIFT. This page references a toy implementation of Scale Invariant Feature Transform algorithms by David Lowe, a leading researcher on SIFT. We were unable to reproduce the successful results claimed by Lowe, but this may be due to our inexperience in the computer vision area.


This follow-up post on GigaOM details the failure of Teleflip, a leading email to text message supplier. Malik reports that Teleflip saw a very limited market for their service. Since our service has an application layer (image recognition) in addition to text-message conveyance, we feel that this is not a direct danger. Additionally, this case reminded us of the relevance of smartphones – email on a cell phone may make text messages relatively obsolete, and it is important for us to build our application to handle both email and text messages.


This paper discusses a new approach that implements efficient lookup of SIFT descriptors over a large database of images. This approach is done by creating a large tree offline using a few million images not necessarily related to the database. Then SIFT descriptors are quantized down the tree creating a series of scores along the way that can be efficiently compared to the database scores to produce results that are relatively accurate.


OpenCV is a library of computer vision ("CV") software which is considered by man to be the best available option for any computer vision-related work. OpenCV was originally developed by IBM, and released as open-source software on Sourceforge. This is the official sourceforge storage, and wiki for the software project. We used this as a starting point for OpenCV development, and for support as we explored its use.
Randerson, J. "Photo recognition software gives location", New Scientist. April 2004
http://www.newscientist.com/article.ns?id=dn4857 Also: http://svr-
www.eng.cam.ac.uk/photobuilder/

New Scientist is a well-known publication for news about new technology, but we
were unable to locate the underlying referenced project. This article documents a
pair of researchers at the University of Cambridge who are building a for-pay service
that recognizes location from a submitted image. As a relatively old article, this piece
presented an interesting lead to a very relevant project, but the seemingly
discontinued nature of the project was discouraging.

2002 Page(s):957 - 960 vol.3

The authors are researchers at the Department of Mechatronics, at Kwangju Institute
of Science and Technology in Korea. This paper outlines an approach for blending
color histogram techniques with edge detection. We used this paper to expand our
knowledge of computer vision techniques, and to consider hybrid approaches.