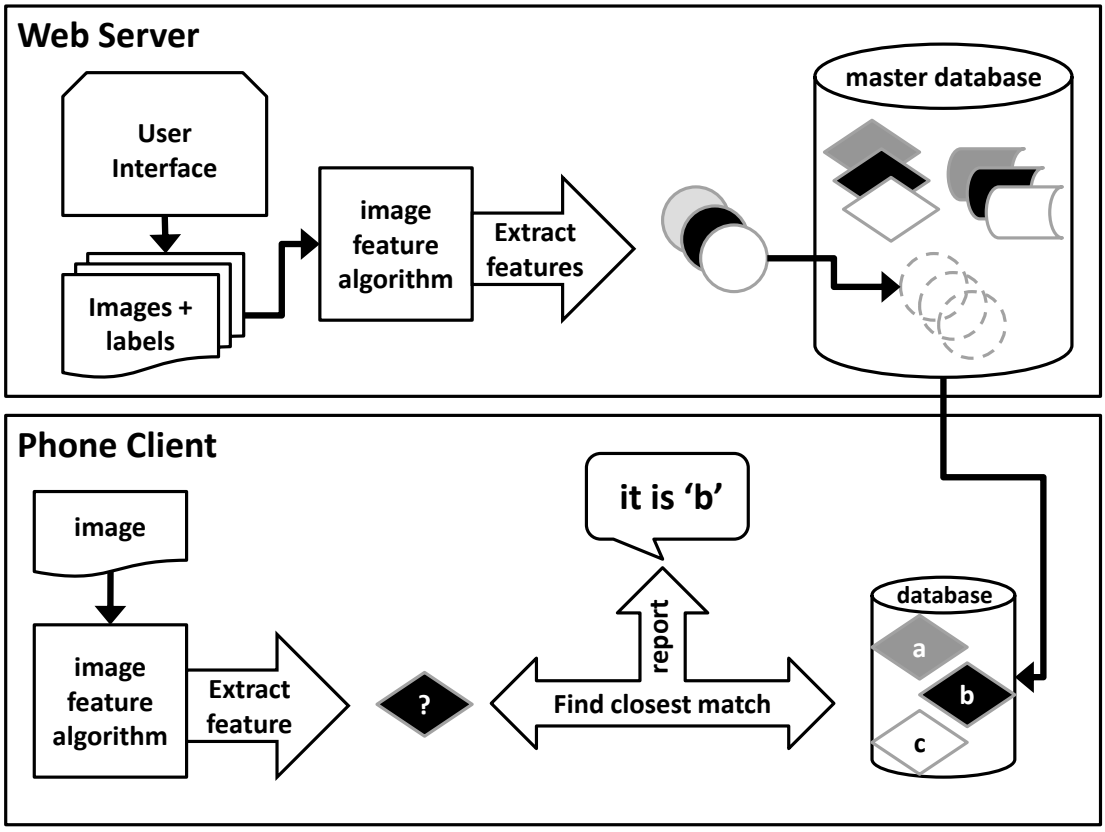
University of Cambridge
Engineering Part IB

Paper 8: Information Engineering Elective

Mobile Computer Vision



Matthew Johnson
May 2009

Applied Computer Vision

Computer Vision is not only an exciting field of research, it is also one of the key new technologies driving innovation in business and industry today. Just as human vision is a fundamental and incredibly rich channel of information which we use for a huge array of activities, advances in computer vision are creating new opportunities for innovation in the way in which we interact with our world through digital devices.

In this lecture, we will focus on ways in which the concepts and techniques you have been exposed to so far in this course are being used today in the field of image recognition and search on embedded devices. More specifically, we are going to look at how we can make mobile phones see.

Impaired Vision

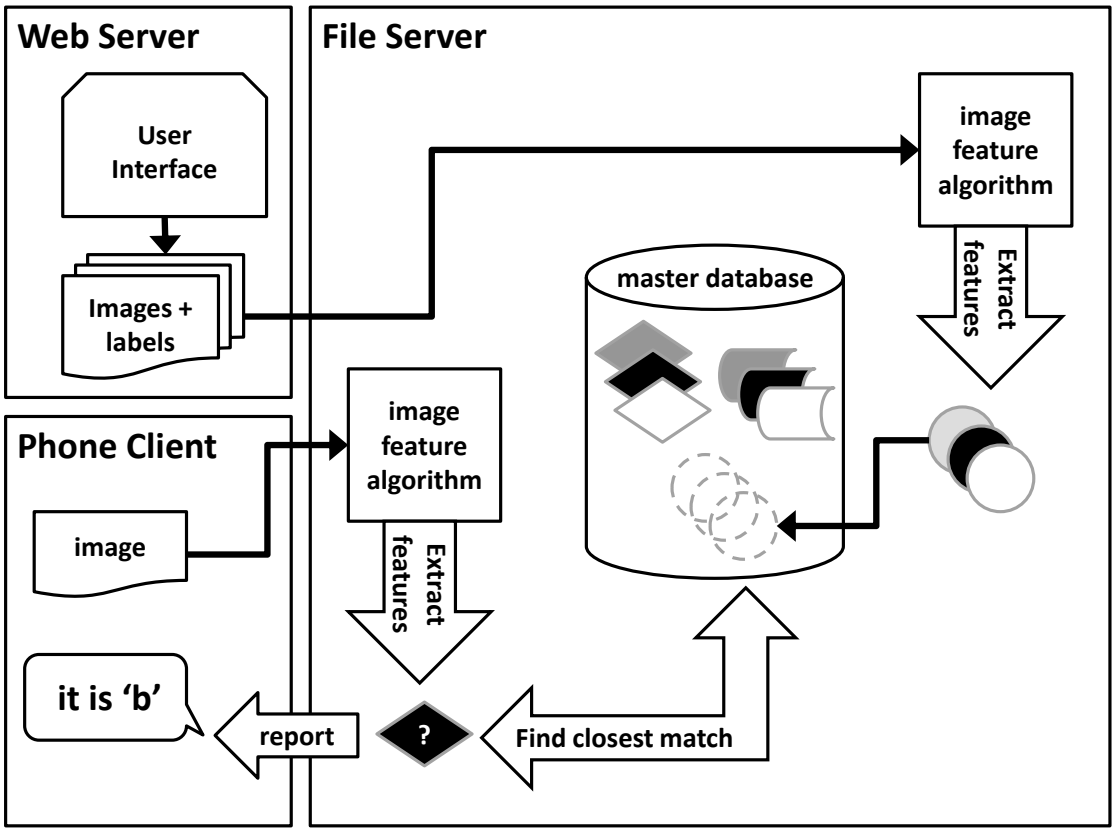
The first thing that becomes clear as one begins to work on a ubiquitous computing device like a mobile phone is that one can no longer take anything for granted. For example, most mobile phones which run the current version of the Symbian Operating System do not have any hardware support for division. This means that any division operation written in Symbian's version of the C++ language is translated in assembly into a long series of other instructions, thus making it incredibly costly.

Here is a partial list of the challenges posed by mobile devices:

- Limited processor hardware (slow floating point operations, lack of functionality)
- Small cache (i.e. limited stack space for procedures)
- Slow access to a small supply of Random Access Memory (RAM)
- Costly disk access
- Arcane versions of known languages

How do we overcome these challenges?

Client/Server Model



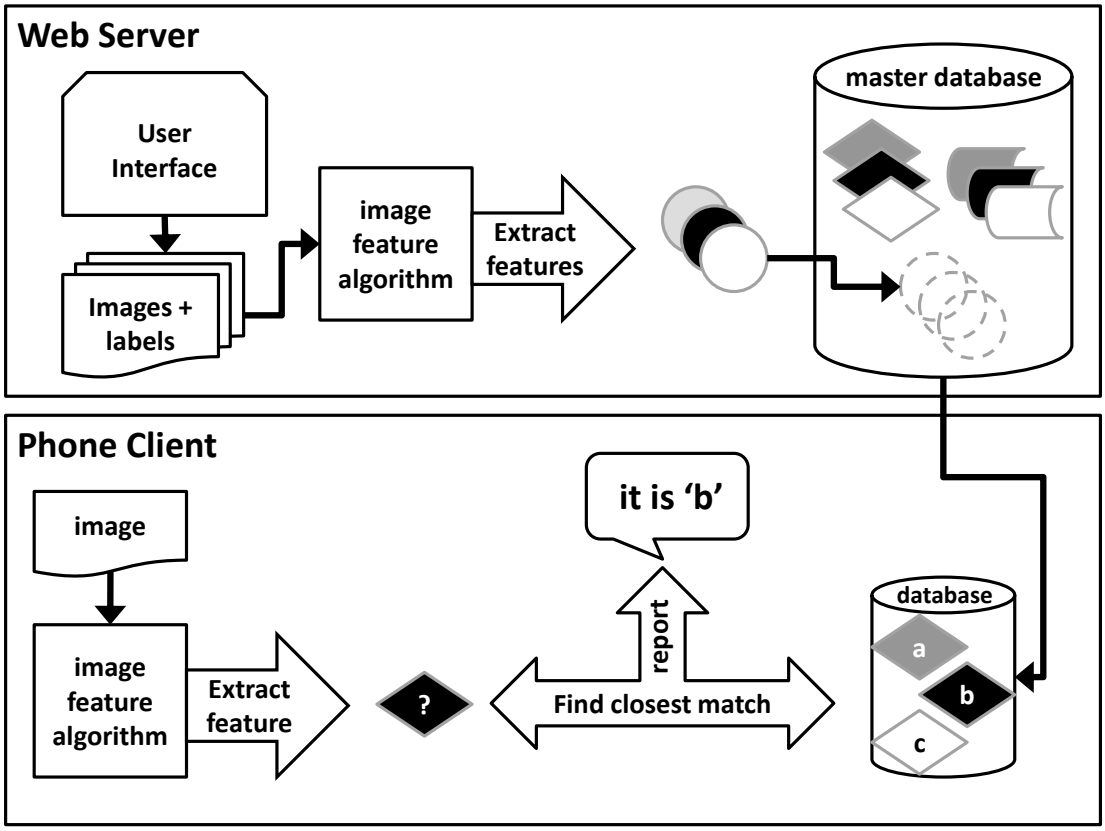
The most common and, at first, only way of overcoming these challenges on mobile devices was to offload any significant computation to a remote server. The system on the phone becomes a "thin" client (meaning that it does no real computation) that sends the image to the server, which computes feature vectors, matches them against a database of known images, and returns the information on the match. In order to add new images to this system, a second system interacts with the server to upload images and labels.

Client/Server Model, cont.

This arrangement solves the problem of the mobile device's limited capabilities, but has several problems of its own:

- If the file server stops functioning, the entire system fails
- If communication to the file server is lost, the entire system fails
- The server(s) have to perform massive amounts of parallel computation and database indexing
- Large quantities of data (i.e. images) require lots of time and bandwidth to transfer
- Significant delay in response to user actions

Smart Clients



An alternative way of approaching this problem is to embrace and overcome the difficulties of performing vision on mobile devices through research into modifications to existing algorithms and entirely new algorithms which are able to solve computer vision problems with the tools available and within the limits imposed. This would be considered a "smart" client model, because the client does some or all of the computation. In this model, the client is able to perform feature vector extraction and matching locally, which means that it must maintain a full or partial copy of the master database on the device.

Smart Clients, cont.

We can see quite clearly how a smart client system, while it has to overcome the challenges of running in a limited environment, also has many benefits:

- If the server malfunctions, clients can still operate with their current database
- Similarly, if communication to the server is lost, the system still operates
- All computation is done by the phones locally, thereby greatly reducing server-based computation
- All communication with the server consists of feature vectors and metadata, much smaller than images
- Immediate response to user actions

Clearly there are great gains to be had, but first we must overcome the limitations imposed by mobile platforms.

System Design

If we are going to achieve Computer Vision on a mobile phone, we must first come to terms with reality. That reality dictates that we have no large constructs in memory, no division, and no floating point numbers. Most of the standard Computer Vision algorithms are simply not available to be used. Instead, we will have to rely on more efficient algorithms, and where those are unavailable, on approximations. A crucial requirement for this task is knowing the literature. We must understand the basic underlying principles of a technique so that we know what can be jettisoned without reducing performance.

We are going to design a visual word based image matching system that only uses integers and can run in real-time on a mobile phone. In order to do this, we will be learning about some new techniques:

SURF A highly efficient SIFT-like interest point and descriptor solution

Pyramid Matching Efficient similarity scores in descriptor and image space

SURF

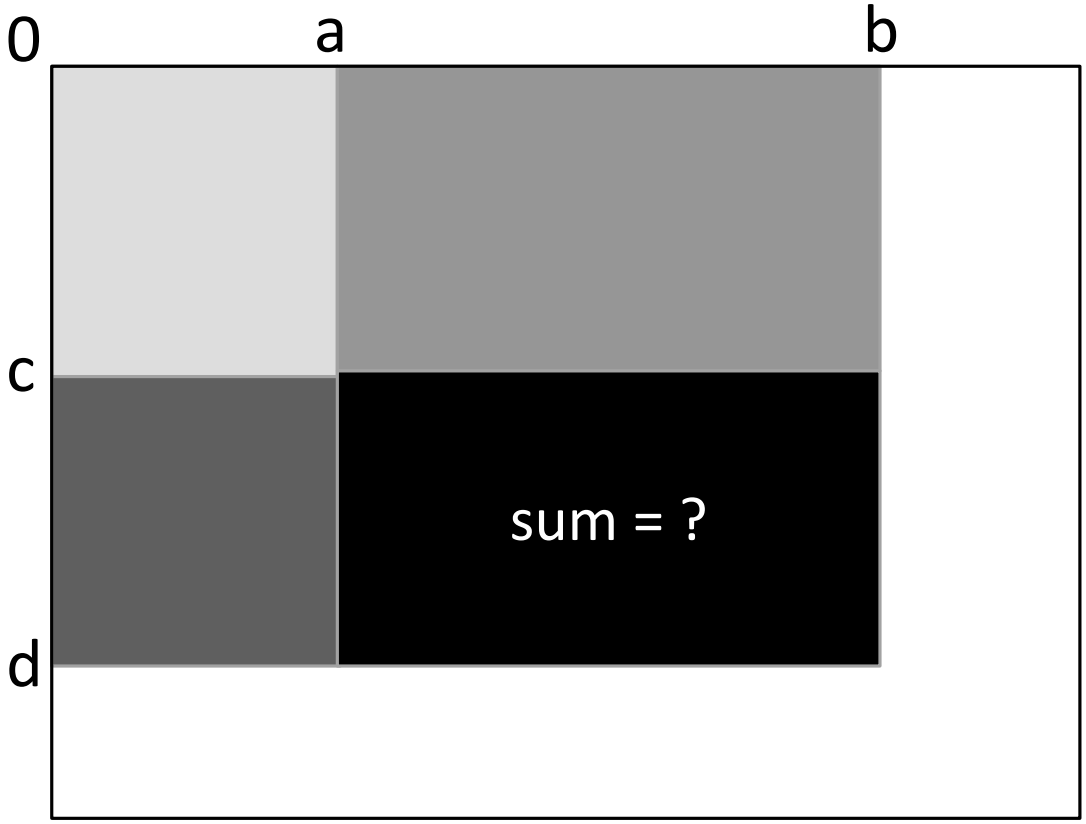
The interestingly named "Speeded Up Robust Features" technique contains some great advantages for us. First, it provides a method of interest point based extraction based on scale-space blob detection. Second, it provides a feature descriptor with incredible expressive power. Finally, it does all of this with integers and without division.

Interest Points The standard method of scale-space blob detection is to find the maxima and minima of the Laplacian of a Gaussian convolved with the image. The most effective way of doing this is to use determinant of the Hessian matrix of the Laplacian. SURF approximates the Laplacian of a Gaussian using simple integer calculations.

Descriptor The SURF descriptor is somewhat similar to SIFT, but instead of creating orientation histograms it concentrates on sums of gradients, which are much faster to compute.

Integral Images

The key component of SURF is the integral image transform. In an integral image, each pixel is equal to the sum of all the pixels to the left and above it in the image.

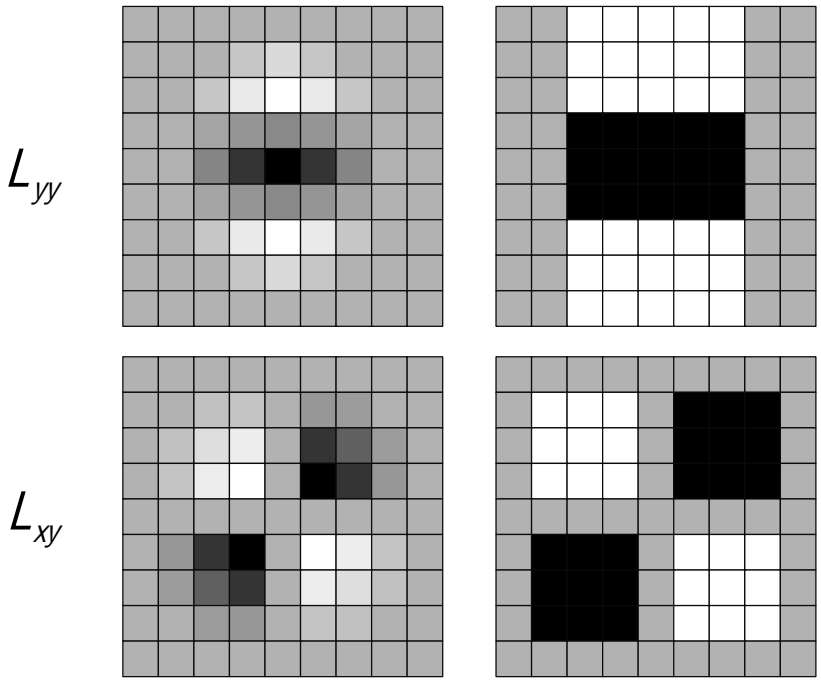


We can use the integral image to find quickly the sum of the pixels in any rectangular area of the image. For example, the sum A of the black rectangle above can be obtained with 3 arithmetic operations and 4 indexes:

$$A = II[a, c] + II[b, d] - II[b, c] - II[a, d]$$

Laplacian Approximation

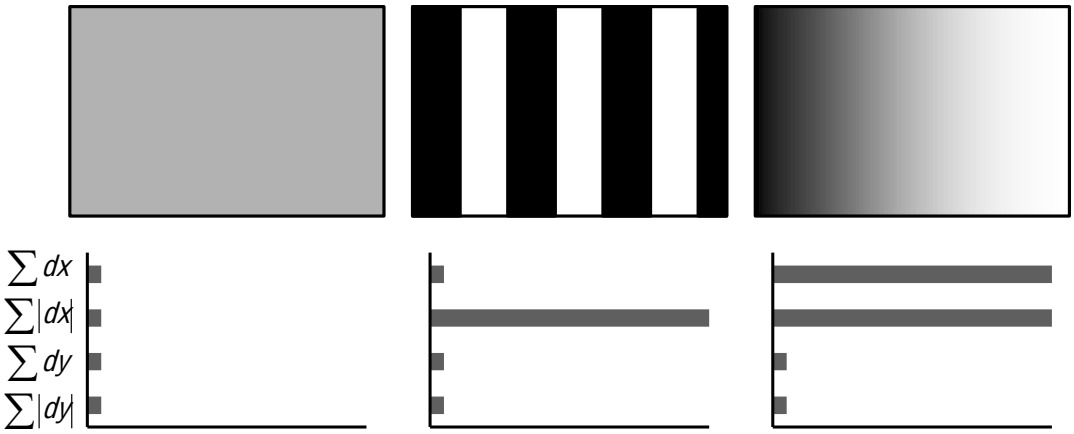
The most common way of finding "blob" interest points is through the use of the Laplacian of the Gaussian convolved with the image. By localising the stable minima and maxima of the function in scale space, we are able to create a strong basis for a visual words based system. The SURF technique approximates the Laplacian as shown below:



On the left are the true function profiles, and on the right the approximations. The approximations are computed using integral images, but lose little of the accuracy of the true function. Also, since the size of the rectangle has no effect on the efficiency of its computation, we can compute this approximation at any scale, thus removing the need for a scale space pyramid.

The SURF Descriptor

The SURF descriptor, computed at all interest points, relies on sums of gradients for its expressive power. While simple, they prove surprisingly effective at describing image textures and patterns, as seen below.



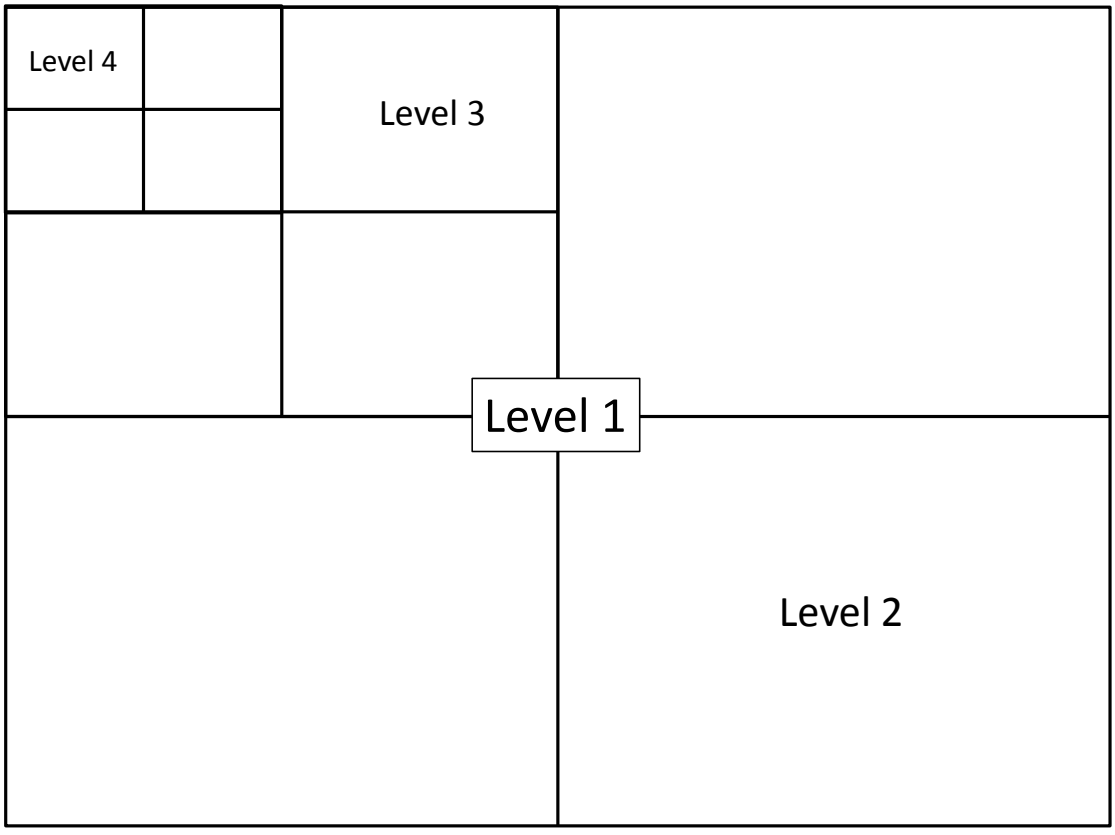
In structure it is very similar to a SIFT descriptor: 16 grid squares, with a descriptor consisting of the four sums shown above computed at each, weighted such that the center is more important than the outside. The gradients are computed using integral images, thus allowing them to be extracted at any scale for the same computational cost.

The key point of SURF is that instead of optimizing the orientation histograms of SIFT, it finds a more efficient way of achieving the same goal.

Pyramid Matching

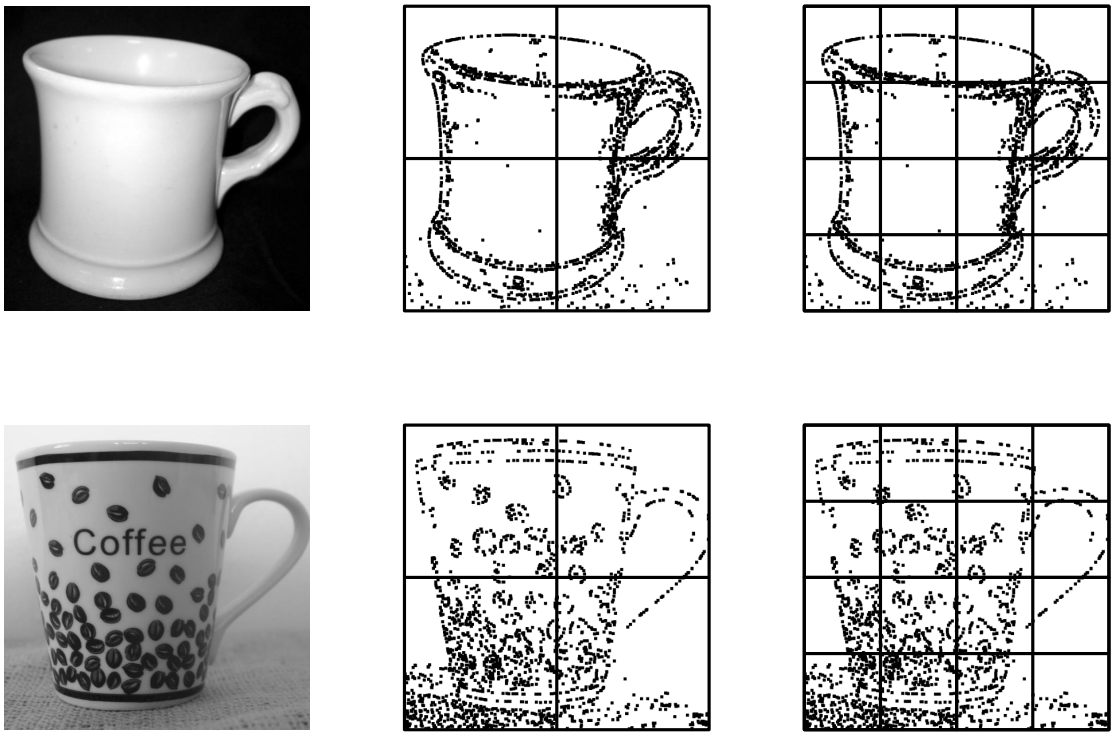
Pyramid matching is an attempt to match not just histograms of visual words, but their locations within the image. A similar concept happens in language. Two sentences with the same words but in different orders will have identical word histograms yet are certainly different. Similarly, if two images have the same visual words but in different locations they should be considered less similar than two images with the same words in the right locations.

In order to achieve this goal, pyramid matching uses what is known as a heirarchical histogram, depicted below.



Pyramid Matching, cont.

At each level, we compare the word histogram in each square to the histogram in the corresponding square of another image. As the level increases, we weight the similarity scores higher.



Above we see two image contours being compared spatially using a pyramid match scheme. The important thing to understand is that as the grid becomes increasingly fine, the correspondence in the number of edgels in each cell increases in meaning, indicating that the two objects have a similar contour and thus may be similar objects.

Pyramid Matching, cont.

The way we compute histogram similarity is through a metric called histogram intersection. The histogram intersection, I, is computed from two histograms A and B as follows:

$$I(A, B) = \sum_{j}^{M} \min(A[j], B[j])$$

Starting at the finest level (e.g. Level 4) we move up the hierarchy and record new matches at each level, N.

$$N_l = I(A_l, B_i) - I(A_{l+1}, B_{l+1})$$

Where A_l is the portion of hierarchical histogram A at level l. The pyramid match is the weighted sum of the N values, as follows:

$$K = \sum_{l}^{L} \omega_l N_l$$

$$\omega_l = \frac{1}{2_{L-l}}$$

The Final System

The final system incorporates these two elements into a visual word matching framework that is able to compare two images very quickly. Here are the steps of the algorithm:

1. Compute the integral image
2. Extract interest points using a Hessian approximation (or, alternatively, use a regular grid)
3. Compute SURF descriptors at each point
4. Assign a word label to each SURF descriptor
5. For each image in the database, compute the pyramid match on a word-per-word basis

Naturally, this is just the overview. There are many problems yet to be solved in the implementation, but by starting with algorithms which are suited to the circumstances at hand, we can be confident in final success.