Summer intership

Creating a building detector

Supervisor : Alan F. Smeaton

Igor Rosenberg

June 16th to September 15th, 2003

Contents

Introduction

This paper presents my summer activity for 2003. I went to Ireland for a research internship of three months, under the supervison of Prof. Alan F. Smeaton. First of all, the reason for this little excursion will be presented, along with the environment I landed in. Then the work produced will be sketched, within two different parts: some little utilities written as helpers to a much bigger and useful software, and my own personal contribution to the research world (a building detector).

1 Context

1.1 Getting there

As a second year student of ENS Cachan, Antenne de Bretagne, I was asked to find an internship abroad as a conclusion to the academic year. The first task was to find a research team to host me. My first move was to narrow down the different topics in which I would like to concentrate. As a consequence to my religious attention to the lectures at my home university, I found an interest in image analysis, because it involves advanced mathematics, as well as program writing and provides visual results that can be objectively judged by the human eye. So after a short interview with Francois Chaumette, I was redirected to Patrick Gros[1]. This meeting proved very helpful. He asked me to think up an even narrower definition of which field I wanted to investigate. Once satisfied, he came up with a list of a dozen of his respected colleagues, who were working in the correct area, and whom he deemed capable of hosting me in satisfiable conditions. Emails were then sent to their teams around the world, with a more personal touch to those he knew better. Very few gave answers. Maybe the french position against the americans' will of supremacy over Iraq was then considered unacceptable. Anyway, once the contact was established, I was given free rein to converse electronically with the possible future supervisor. I was to bargain my housing conditions and salary, as well as the subject. Alan F. Smeaton [2] appeared as the most enticing, and was chosen as my final decision. So once both parts had agreed by contract, accomodation was arranged for me by the hosting team. The subject was left unspecified, and considered as the first task of the intership. A really cool salary was obtained later on during the summer.

1.2 The CDVP in Dublin City University

The Centre for Digital Video Processing is a cross-disciplinary research centre and a collaboration between the School of Computing and the School of Electronic Engineering at Dublin City University. Their mission is to research and develop techniques and tools to automatically analyse and index digital video information and allow content-based operations such as browsing, searching, alerting, filtering and summarisation.

I worked in the second floor lab of the computing applications building. I had a desk in the common postgrad room, along with other postgraduates. I was entitled to use all the computing resources available, even though I only used three different computers and the printer, but there were more ready to use if needed. I shared the lab with seven other people, mostly PhD students

[1] http://www.irisa.fr/texmex/people/gros/index_fr.htm

[2] http://www.computing.dcu.ie/~asmeaton/

and postdocs. Thanks to the proximity of our desks, I was taught a great many things by this cheerful group of people...

The lab is involved in several project, listed on their home page. What took most of people's time when I was around was their thesis, obviously, and TREC. TREC[3] is mainly a testbed providing data sets and tasks to be fulfilled. Teams around the world are invited to compete in any, and the results are evaluated before the conference. It allows for an objective evaluation of different approaches to well-specified problems.

1.3 People

Here is a short list of the people with whom I had the biggest interaction. They can be reached through the web page `http://www.cdvp.dcu.ie/members.html`. My stay was such a pleasure thanks to their care. They have all my gratitude.

- Mr Neil O'Hare will start his PhD this year. His current work mainly consists in detecting anchor person shots in news video. He asked me to write little utilities to transform some file format, and introduced me to Support Vector Machines.
- Mr. Tomasz Adamek is a PhD STUDENT, since 2001, in the school of electronic engineering. He is currently working on object detection. I asked what was feasible in image analysis, and which tools I could use. I appreciated his warnings about never getting the shape expected, and how even low-level features were not always satisfying.
- Dr. Noel O'Connor gave the guidelines to what my own research should be. He gave me a very good start without imposing his ideas.
- Mr. Kieran Mc Donald will finish his PhD this year. He was most kind to answer to my never ceasing questions about the framework of the fischlár software. He was the one who asked me to write some little tools for this year's TREC. He also checked on how I was doing.
- Dr. Cathal Gurrin (postdoctoral fellow) asked me to import the Closed Caption. He also took care of my internship when Alan was away and helped me get integrated in the team.

2 Different modules used by Trec

As a junior member of the CDVP, my partipation was required to write small tools for the TREC 2003 search task[4]. The input files, identical to all teams around the world, were not always in the format defined within the lab. So many conversions were required, the usual work consisting in regrouping information from several sources and combining them in main MPEG7 description of the given video.

A common interface was ready to use for every different process. Each tool should have a `process` and a `configure` method. It should be added in the xml configuration of the complete software, and then everything should be rebuilt, and run.

[3] http://www-nlpir.nist.gov/projects/trecvid/

[4] http://www-nlpir.nist.gov/projects/tv2003/tv2003.html

2.1 Alignment

A problem that arose in the beginning of the team's work on Trec was the wrong time alignment of the sound track of the video. It was discovered that there was a difference between the motion pictures and the sound track. This time difference was specified for each file, but as it had been done manually, it contained errors. So the first alignment programs used this wrong table, before the error was discovered. Then a second alignment was issued, but in another format. Nevertheless, I created a small java program (`AddDelay`) to shift the times read in some files to make them correspond to the correct moment.

The TREC guidelines often change, unluckily. At the end of August, the officials asked not to use the original ASR that this work was based upon, but use another one already aligned. So all this work can been thrown in the bin...

2.2 Automatique speech recognition

The first tool I wrote (`ImportTrecAsr`) was very simple. It allowed me to get back into the JAVA language. Files containing ASR relative to a given video were produced, in an alien format. I wrote a simple parser that reads such a file, and inserts the words in the MPEG7 descriptor, under the correct shot. A very simple alignement was used: since the asr files contain a time stamp for every asr word, each shot's ending time was computed and all the asr words prior to that time and not already used were added to that particular shot's description. That was done iteratively for all the shots of the video.

```
<DOCSET type=ASRTEXT fileid=19980628_1130_1200_CNN_HDL collect_date=19980628_1130 collect_src=CNN
           src_lang=ENGLISH content_lang=NATIVE proc_remarks="Byblos English ASR">
<X Bsec=0.00 Dur=33.42 Conf=NA>
<W recid=1 Bsec=33.42 Dur=0.16 Clust=NA Conf=NA> AND
<W recid=2 Bsec=33.58 Dur=0.22 Clust=NA Conf=NA> H.
<W recid=3 Bsec=33.80 Dur=0.08 Clust=NA Conf=NA> I.
<W recid=4 Bsec=33.88 Dur=0.16 Clust=NA Conf=NA> V.
<W recid=5 Bsec=34.04 Dur=0.66 Clust=NA Conf=NA> POSITIVE
<X Bsec=34.72 Dur=1.00 Conf=NA>
<W recid=6 Bsec=35.72 Dur=0.16 Clust=NA Conf=NA> TO
<W recid=7 Bsec=35.88 Dur=0.46 Clust=NA Conf=NA> IMPORTANT
<W recid=8 Bsec=36.34 Dur=0.21 Clust=NA Conf=NA> NEW
<W recid=9 Bsec=36.55 Dur=0.52 Clust=NA Conf=NA> STUDIES
<W recid=10 Bsec=37.13 Dur=0.26 Clust=NA Conf=NA> SHOW
<W recid=11 Bsec=37.39 Dur=0.13 Clust=NA Conf=NA> HOW
<W recid=12 Bsec=37.52 Dur=0.23 Clust=NA Conf=NA> WOMEN
<W recid=13 Bsec=37.75 Dur=0.18 Clust=NA Conf=NA> CAN
<W recid=14 Bsec=37.93 Dur=0.54 Clust=NA Conf=NA> PROTECT
<W recid=15 Bsec=38.49 Dur=0.17 Clust=NA Conf=NA> THEIR
<W recid=16 Bsec=38.66 Dur=0.53 Clust=NA Conf=NA> BABIES
<W recid=17 Bsec=39.19 Dur=0.15 Clust=NA Conf=NA> FROM
<W recid=18 Bsec=39.34 Dur=0.06 Clust=NA Conf=NA> THE
<W recid=19 Bsec=39.42 Dur=0.56 Clust=NA Conf=NA> VIRUS
```

Example asr file. $Bsec =$ is the time stamp.

2.3 Moving images

My next task (implemented in `TrecKeyFrames`) was to rename the keyframes that had been supplied by TREC so they could be used with existing DCU tools like the face detection tools Indeed, all the 68000 jpegs corresponding to the whole test collection were generated in the same directory. The images were originally named as shot<video nb>_<shot nb¿_<type of video>.jpg. These names were to be transformed into
<framenumber>.jpg, and moved to the correct directory, parameter of the tool. The formula

$$framenumber = time * framespersecond$$

was used to know which framenumber to attribute to the keyframe, the time of every shot being read from the MPEG7 description. The standard value of 25 frames per second was used when this information did not appear in the input.

2.4 Width and height

Another little tool was written to complete the MPEG7 description associated with each keyframe - such as adding correct width and height format information. The algorithm is pretty straightforward: for every shot, open the keyframe image, read width and height, and add them to the shot block of the descriptor. Since so little was done, I simply extrended `TrecKeyFrames`.

2.5 Rescale images

A need to have the keyframes in another size appeared. Indeed, in the fischlár interface, several sizes for pictures are defined, depending on the place they appear. This is used to highlight the shots that are considered relevant. So a Tool called `TrecThumbNails` was created. It reads the whole MPEG7 description, and for each shot, creates a keyframe (prefixed by `thumb_`) of the size specified in the input, by scaling the original keyframe to the appropriate dimensions, and then stores information about the new thumbnail back into the MPEG7 description.

2.6 Import closed caption

The last tool I wrote (`ImportTrecClosedCaptions`) is a trifle more complicated. The ASR gives precious information, but is not 100 % accurate. Another source is the closed captions. For the news programs used as data, the transcription of what is said is added as hidden subtitles, the **closed captions**. This is very accurate, since it is done manually for every news story. Unluckily, the closed captions supplied by TREC have no timing information. The words are separated from each other, but there are absolutely no timestamps. So a way had to be devised to use this very useful information, and to add it to the correct shots, using the ASR as reference. The problem is the following: given two sequences of words, find the best one-to-one matching so that

- A match consists of two similar words: *Mountain.* and *MOUNTAIN* must match
- The matches do not cross: $x > y \implies match(x) > match(y)$
  where $x > y \Leftrightarrow x$ appears after y
- The matching is maximum: as many words as possible are matched

This is very similar to the 1999 exam[5] proposed by the ENS. It exhibits a way of constructing a best match, in complexity $O(M * N)$ where M, N are the sizes of the two inputs. Since I had already worked on it, I got started right away. The implementation proved a little tricky, but was finally put right. So the closed captions were added to the shot description very accurately.

Nonetheless, improvements can be made:

- the match on words is based on exact letter-to-letter match. For example *boot* and *boots* do not match. Heuristics could be devised to deal with the english language constructions like *s* to plurals, or *ed* to verbs, so that more words could be recognized as equal.
- the exact matching method has not been investigated too deeply. For example, consider the following input: *red and blue* **hand** *green* (ASR) and *red and blue and green* (CC). *and* appears twice in the CC. It is matched properly, because the colors around it are regognized properly, but if the input was *... and .... hand ...* (ASR) and *... and*$_1$ *... and*$_2$ *....* (CC), which *and* in the closed caption would be used for the match?

[5] http://www.lsv.ens-cachan.fr/dptinfo/concours/annales/cor99-I.ps - part III

- Unregognized border words are not treated fairly. For example, consider the following ASR:

  1. *Here is the weather fur cast* (Shot 1)
  2. *of the you West.* (Shot 2)

  And the corresponding CC: *Here is the weather forecast for the US.* The words underlined can not be matched. But the CC should still be assigned to a shot. My implementation always stops the shot at a match, so the words not matched are assigned to the next shot. Another way could be to assign half of the border words not recognized to each shot, or maybe use advanced word regognition on these few stray words. This would need to be investigated more thoroughly, which I didn't do, due to lack of time.

These are not very important issues, since the matching is very successful already. When comparing visually both texts for every shot, very few big errors are reported. Very big chunks are accurately matched, and then for a handfull of shots, the matching is wrong, but catches up again later on. This is due to advertisement breaks which are picked up by the ASR, while no closed captions appear. It also happens when the topic many difficult words are used. Difficult words are words not in the ASR dictionary. Since the words are not recognized properly, no matching method could pick up the good matches. The objective precision is hard to judge. One method is to manually count the errors

$$precision = \frac{number\ of\ matches}{number\ of\ ASR\ words}$$

which gives an average precision of 0.483. This does not account well for all the words that are correctly placed, because between two matches, but not recognized. And telling whether a match is correct (as in the and_1 example above) would require a ground truth which was not provided... I manually counted the errors on a file not used in the development, and then I found

$$precision = \frac{number\ of\ shots\ with\ correct\ matching}{number\ of\ shots} = \frac{282}{315} = 0.895$$

a shot has correct matching when most of the asr and cc words match. An example of a correct shot is

<ASR> and yeah i can do this i get mine and education
<CC> i can't do that, so i get more of an education.

It is a little biased by the information around it, because both the matches before and after were correct, so this one, even though only 5 words match (out of 12), is considered ok.

3 Real research

During our first interviews, Alan F. Smeaton insisted that he wanted me to be doing two things, as he is used to do with his interns. On one hand he wanted me to be integrated in the lab, and a good way of achieving that was to be writing utilities for the other research people. On the other hand, he wished me to come up with something more personal, that would be entirely my doing. So we came up with a global idea of what I could do as research.

3.1 The problem

One of the numerous problems in image analysis is to know what elements are part of a picture. For example, recognizing trees, people, cars, tanks or planes would help in a great many applications. In TREC, one of the features to be extracted is landscape vs cityscape: is the picture taken outside in the countryside, or is it in a city? A way of saying it's in a city is to recognize buildings. Buildings stand out as the common factor of most towns, and have common features that make them good candidate for computer recognition. So I agreed with my supervisor to try to make a system, which given a picture as input, would give a probability that the picture contains a building.

No assumptions at all were made on the image: night/day, sky visible or not, close-up/far-away, noisy/clean/blured, ...

3.2 Litterature survey

I read a few articles before starting my own research. These were the articles written by people who had done TREC on previous years. I found them by going to the web pages of the team involved in the feature extraction task, and having a look at their publications, as well as by looking at the papers for the Trec 2002 conference. The conclusion I got to, after reading them, was that very little was known on how to solve such a problem, and nothing was said as to which method would work better than others. The already existing detectors seem to be having at most a precision of 40%.

3.3 Ideas

After one of the weekly meetings, I had a talk with Dr. Noel O'Connor[6], from the Engineering School. We discussed the approach to come up with, and what features could be used. He thought of using:

1. Region segmentation of image. Extracting parts of the image that have a common appearance could take the search of a big picture down to examining smaller regions.

2. Dominant color. Buildings usually don't have natural colors. They are red, blue, white, but hardly ever brown or yellow or green.

3. Texture homogenity. A building is a human structure and obeys to our conception of order: they look regular. For example, windows are evenly spaced. The texture could be analysed to grasp the regularity of different areas of the picture.

4. Frequency analysis. The frequency can be interpreted as a regularity of the whole input. Since a picture of a building should be fairly regular, this should show in it's frequency function. The Fast Fourrier Transform was thought of.

5. Edge histogram. Buildings are very vertical structures. In an edge picture, they appear like a set of more or less pararallel lines. So it was suggested that a vertical and horizontal lines count would maybe differentiate the buildings from other natural structures, which would have more diagonal components.

[6] http://www.eeng.dcu.ie/~oconnorn/

6. Support Vector Machine (SVM). Once several features have been extracted from pictures, the trick is to find a way of weighing them to end up with a final confidence measure. An SVM does exactly that. Given enough learning material, it can be taught to split it's input space into similar regions.

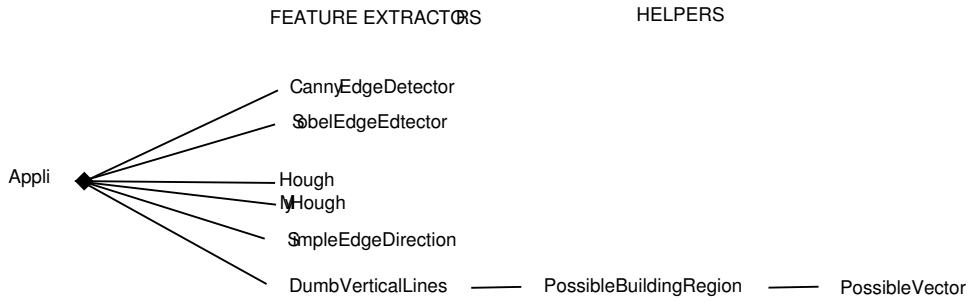
3.4 Implementation

The application was written in JAVA and depends only on the standard JAVA libraries, so it can be ported to any plateform. The project was developed under Linux, and was tested successfully under Microsoft Windows 2000.

3.4.1 Structure

The program is capable of displaying in a frame what the features detected look like on the image, by highlighting the relevant region. With the -X option, this is not sent to a new window on the desktop, but rather to the standard input. This allows to get more information than just yes/no, in a reusable way. This was used extensively during the development. But the true use of the tool is the procedure `Appli::test_contains_building(String)`. This can be called externally, and returns a boolean that says whether a building is recognized in the input.

The internal structure looks like this:



There are several tools used in the building detector application. They are either used one after the other (edge detecting is used by all), or in concurrency (eg: the score assigned to the region is function of all the scores obtained).

3.5 Execution

The program executes:

- **Appli(source files array)** calls the building detector on each of the parameter names. Also creates the host frame, whether virtual (with the -X option), or real (in which traces of the execution appear on screen).
- **testContainsBuilding(file name)** runs the different tools on the image corresponding to the string given. It returns a boolean, false meaning that no building was found.

- **applySobel** from the given image, extracts its contours as a grey-level image. The Canny edge detector is also implemented, and yields thinner edges. I decided empirically that I would rather have a lot of contour points, so Sobel was chosen.
- **applyDumbVerticalLines** Regions of interest are extracted here as being situated around vertical edge point lines. An instance of the class `DumVertical` is created, and allows the image to be parsed with more or less vertical lines (a tolerance is defined in `Const.java`). The lines that hit the most edge pixels are saved as belonging (probably) to buildings.
- **applyBuildingAroundVerticalsExtractor** Around each of the previous lines, a region is expanded. The confidence value is worked out.
- **makePossibleBuildingRegion** From a vertical segment (doesn't have to start from the first pixel on the bottom line), uses a heuristic to iteratively widen the region around it. This is done (in the latest version) by examining bunches of N pixels, and comparing their average hue and saturation to those extracted from the corresponding bunch on the original vertical line.
- **applyGlobalFormulae** Combines the confidence measure of every extracted region, and yields a global confidence measure for the whole image.

3.5.1 Different measures

Once a score has been assigned to every important region of the given image, a global score is to be EXTRACTED/DEVISED. Several formulae, from trivial ones to more subtle ones, may be applied :

- **Max:** $score = \max_i\{score_i\}$
- **Weighted mean:** $score = \frac{\Sigma_i score_i * area_i}{\Sigma_i area_i}$
- **Position dependant:** $score = \Sigma \frac{score_i}{distance_i + 1}$
- **Average of best:** $score = \frac{best\ score + 2^{nd}\ best\ score + 3^{rd}\ best\ score}{3}$

The last method is currently used

3.6 Evaluation

In the last week of the intership, I finally managed to get the TREC 2002 development collection. It states for every keyframe whether it is a cityscape, or a landscape. This was done manually for the whole collection, so these pictures could be fed in as test. They provide ground truth, and the precision of my tool can therefore be precisely measured. This final testing was done on the last day, using `script/run.sh`, and provided the following results:

number of test images:	268	
correct building detections:	5	
correct building negations:	188	
false positives:	12	(tool says there is a building, but none appears)
false negatives:	62	(says there is no building, when one can be seen)
precision:	29.41 %	
recall:	7.46%	

This shows improvements are necessary. The tool cannot be used as is, and provides noise more than anything.

$$precision = \frac{number\ correctly\ returned}{total\ returned}$$

$$recall = \frac{number\ correctly\ returned}{total\ correct\ in\ collection}$$

The main problem lies in the detection once a region is selected. I'm sure that with a little tweaking, much better detection could be achieved.

3.7 Thresholds

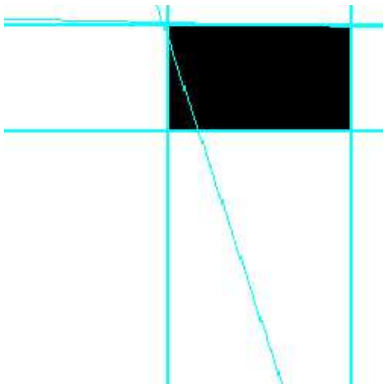
I thought, in the beginning, that only a little tuning would make the tool useful and precise. As I got more and more code written. more and more constants appeared as thresholds to different algorithms. Even simpleEdgeDirection needs a threshold: when is an edge pixel relevant? $value > 0$ or $value > 128$? So here's a list of these, and were they appear. I haven't had time to try out their different values. It would have been nice anyway to be able to hide them, or at least get them by trial and error, but time flew by so fast...

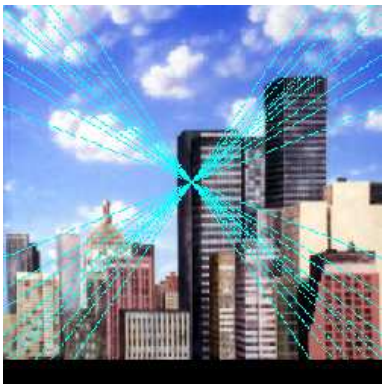
- $X\_Y\_RATIO = \frac{1}{20}$: this is used as a step to get an angle not totally vertical.
- $BUNCH = 10$ used in `PossibleBuildingRegionExpandByBunchColor`, it is the number of pixels in each bunch
- $MAX\_HUE\_DIFFERENCE = 0.1$ colors are considered similar if their $hue^2 + saturation^2$ difference is less than this bound
- $NB\_VERTICALS\_TO\_EXTRACT = 8$ Only consider this many regions in the image
- $MIN\_EDGE\_LENGTH = 10$ All regions must be based on segments at least this long
- $SMOOTHING\_SCALE = 8$ when trying to expand regions, I thought working on the original image would give bad results. So at this stage, the image is smoothed with a window of this size. I didn't really see if smoothing helped at all. Maybe it doesn't...
- $FIXED\_EXPANSION = \frac{1}{4}1$ is only used for `PossibleBuildingRegionTrivial`. So don't consider this.

3.8 Discarded

Other things have been implemented and were finally rejected:

- I spent quite some time getting the Hough Transform [1, 2] to work. This is a method to extract lines from an image by accumulating the lines as coordinates, where a line desribed by $xcos\theta + ysin\theta = \rho$ is stored as (θ, ρ). This method works very well on internet demos, and mine works wonderfully well on simple shapes like rectangles. However hard I tried, I never managed to get it to give interesting information on more complicated images. It would draw lots of lines starting from the centre, but nothing else. So after two weeks of struggling, I gave it up to replace it with the `DumbVertical` extractor.





As you can see, the results were promising on simple shapes, and very deceiving on more complicated pictures.

- For the edge pixels detector, I chose Sobel. Canny was discarded because I thought thick edges were better than thin ones. This has not been proven, though.
- I found another utility that promised to find lines as well, using a kernel. But it needed too many parameters that had to be tuned. So it was also thrown away.

3.9 Time flies by

As the internship lasted for only three months, things were sketched but never got near to be finished. Here's a list of what could be added to the project.

- An SVM [3]. The idea of guessing a function to return the score assigned to a picture is not satisfying. It takes too much into account what people think is representative of a building, and not enough what a building really is. If enough relevant features were to be extracted from an image, one way of combining them would be to use a Support Vector Machine. An SVM is able to split space into regions, making its own rules from the learning set it is taught with. Of course, human interaction is very important there as well, but the exact line between regions is chosen by the machine, rather than by some dodgy trial and error.

4 Conclusion

My second experience in the research world was very interesting. I did not get enough contact with my supervisor though. I have learnt that I must be more active in my search for human interaction, as it is a very important part of the job. Having input from other people allows for a wider or simply different view of the problems and generates better thinking. The simple part of what I was asked to do (the different modules) was done to my satisfaction. I got very enthusiastic feedback from Cathal about the Closed Caption matching, which delighted me. On the other hand the building detector would need either a totally different approach or a very strict cleaning up, because the results given are nowhere near satisfactory. But at least it shows the task I was given was not as simple as one might think at first hand.

No matter what you do or say
Time will always tick away

5 Dictionary

Here is a small list of terms used throughout this report, and their meaning

- **story** . News video are segmented into stories, an a sequence of shots in which the topic remains unchanged.
- **shot** . A shot is a sequence of images in a video in which the camera does not jump, or in which no cuts or fades appear.
- **keyframe** . For every shot, a keyframe is an image selected from the set of images making up the shot. It is representaive of the information of the shot.
- **ASR** . Automatic Speech Recognition. Extracted from the sound track, words are written to be used in search engines. It is not very accurate.
- **Closed Caption** . Transcription of the audio track. It is done manually, so is the is a perfect copy of what is said.

References

[1] Hough Transform, `http://www.cs.technion.ac.il/Labs/Isl/Project/Projects_done/VisionClasses/Vision_1998/Hough/hough.html`; accessed August 23, 2003.

[2] Hough Transform, `http://www.dai.ed.ac.uk/HIPR2/hough.htm\#1`; accessed August 23, 2003.

[3] SVM^{light} : C implementation and tutorial to Support Vector Machines, `http://svmlight.joachims.org/`; accessed August 23, 2003.

[4] Michael S. Lew, "Principles of Visual Information Retrieval," *Springer-Verlag*, Advances in Pattern Recognition, ISBN 1-85233-381-2

[5] Ciarán Ó Conaire, *MPEG7 Feature Extraction for Advanced Content Retrieval*, Degree in engineering in telecommunications, supervised by Dr N. O'Connor, 2003

[6] Yining Deng, B. S. Manjunath, Charles Kenney, Michael S. Moore, Hyundoo Shin *An Efficient Color Representation for Image retrieval*, IEEE Transactions on Image processing, Vol 10, no 1, Jan 2001

[7] "Detection of Linear Features in SAR Images: Application to Road Network Extraction," Florence Tupin, Henri Maître, Jean-Francois Mangin, Jean-Marie Nicolas, Eugene Pechersky, IEEE Transactions on Geo science and remote sensing, vol 36, no 2, March 1998