Bilkent University

Department of Computer Engineering

Senior Design Project

IRIS

Video Analysis and Retrieval System

Elif Onalan, Korhan Bircan, Fatih Koca, Ozan Dincer

Committee Members:

Pinar Duygulu (Supervisor)

Selim Aksoy (Co-supervisor)

Ozgur Ulusoy

Progress (Final) Report

May 6, 2005

Table Of Contents:

1. Introduction:

IRIS is a video retrieval system that is created for the need in the video retrieving systems area. The current systems become insufficient day-by-day as the archives grow very fast and become huge. They are insufficient because they search the data by queries from the archives that are prepared manually by attaching a text value for annotations of a video entry while putting it to the archive. Then the system searches the queried information from these annotations and matches the queried information with the videos that have correct annotations and only the exact matching word is being searched within the annotations [1].

Huge archives made this process inefficient. However, the goal of IRIS is to make the video archives available to a great variety of users in an efficient, effective and organized way. It is basically a content based video retrieval system. It searches lots of shots in a dataset to plug out the queried information. It has better retrieval results by multi-modality that is by integrating different modalities such as visual information, speech and text [7]. Moreover, it provides services for detecting commercials and faces. The search and retrieval is based on both visual information and text unlike the old systems that is based only on text [4]. IRIS also makes the search in video archives in more semantic way by making use of classifiers.

This report is the final report that gives a basic idea of how we have produced IRIS, what stages we've been through and what services we have used in order to develop this system. First of all in section 2, we will mention about our dataset, what we had in hand and what was used previously to produce this data. In section 3 we will talk about the system overview by mentioning about the two fundamental modules in our project. These two modules are low level modules and mid level modules which are mentioned in section 3.1 and 3.2.

Low level modules are processing the basic properties of this data such as the text –which is the speech transcript and annotations- and the low level visual features. Therefore we have 2 sub-modules in this section which are text extraction and low level visual feature extraction (as can be seen in figure 3). Text extraction will be explained in detail in section 3.1.1 and low level visual feature extraction will be explained in 3.1.2.

After finishing the low level modules we have moved our work into an upper semantic level which is the mid-level. In this level we are intensely using the low level module outputs and the face recognition outputs that were provided to us. We have 2 sub-modules in mid-level which are face recognition and scene classification (as can be seen in figure 12). In face recognition of mid level modules we have used the face detector outputs of a face recognition software provided to us. This will be mentioned in section 3.2.1. Then we are classifying scenes which are for now basically distinguishing the commercials and sports news. This module is using both low level visual features and text extraction and will be explained in section 3.2.2.

In section 4 we are talking about our database, which is the base of all. All of the outputs of these preprocessing modules and all of the information about the data we received are put in the database. This section contains all of the tables and some short definitions about their functionalities.In section 5 we are talking about some of our crucial classes mainly text extraction, classification and database classes. In section 6 we are demonstrating the scenarios on GUI and lastly in Appendix you can find all of the specific definitions, abbreviations and acronyms that are used in this document.

2. Dataset:

Content-based video retrieval system is an integration of image and audio retrieval systems. We needed a video archive which is rather useful for many areas. Therefore we have obtained the data from TRECVID [8] 2004 at NIST. Our archive is composed of ABC and CNN televisions' news video archive from the year 1998. Such a news archive is useful for reporters, journalists, commercial agencies, social scientists-especially the ones interested in history-, students and basically people who are searching for information in videos. There are 254 videos and generally their length is about 35-45 minutes. These videos contain news, sports news, weather news and commercials too. Different shots from the videos are given in figure 1 for you to have a better understanding about the content of these videos.

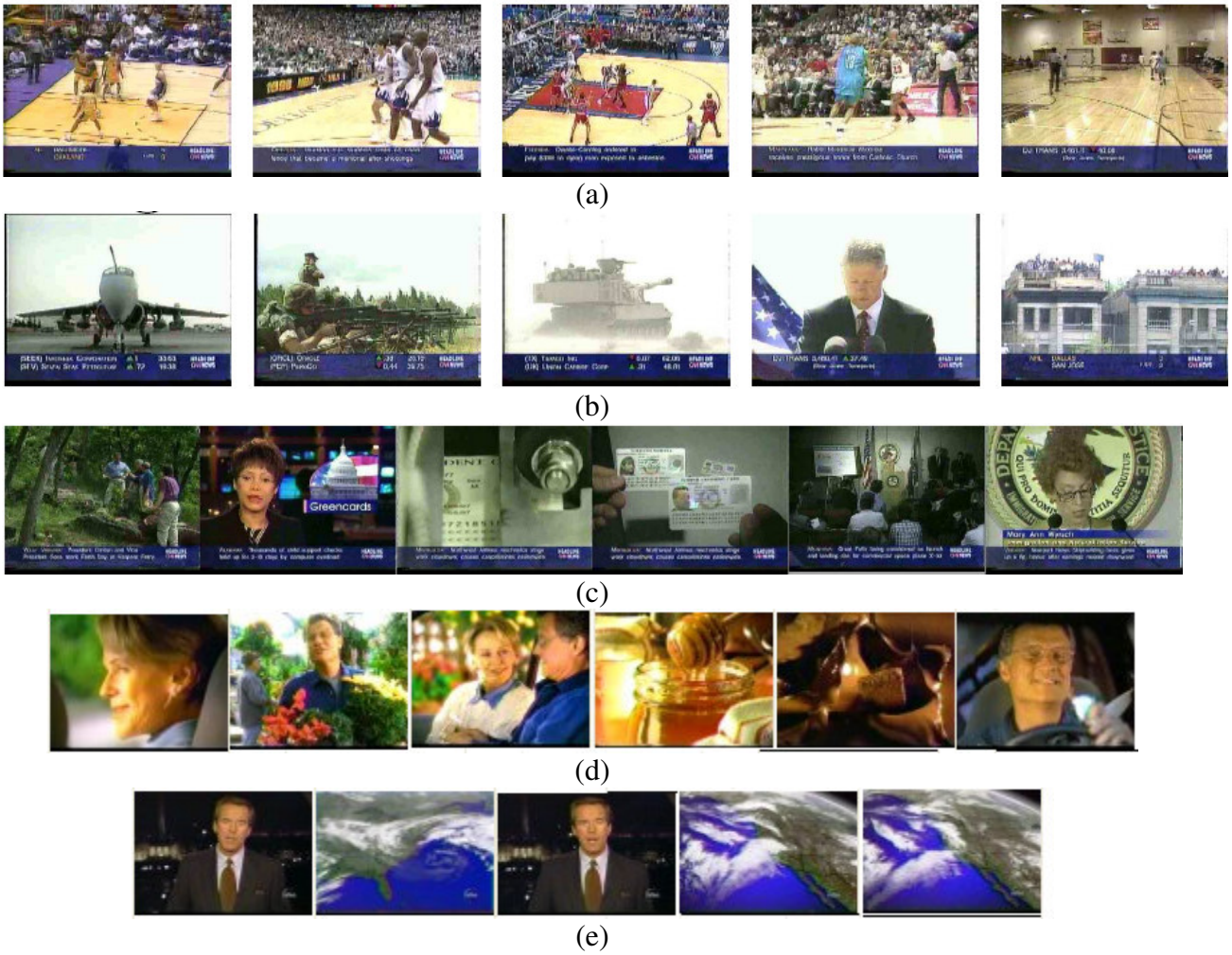


Figure 1: a) a shot sequence from sports news, b) an sample news sequence, c) a sample news sequence with anchor in it, d) a sample sequence from commercials, e) a sample from weather news

We obtained some extra detailed information that we made use of a lot in this project. This information contained 3 different aspects of the videos. This information are the general shot information for each shot in each video, the speech transcript file with confidence values and the annotation which was prepared by approximately 100 groups of people manually by watching the videos and considering the key frames. Before explaining each, we needed to mention that we have aligned all of these data in one format where shot is the basis such as all information was mapped to each shot by VideoID and ShotID.

⇒ The video's shot information has the shot name, shot's beginning in time, shot's beginning in terms of frame numbers, shot's duration in time and shot's duration in terms of frame numbers, the key frame name in the shot and the key frame number of the shot. The data we received was in the following format:

```
shot1_1 0 0 8708 261 shot1_1_RKF.jpg 4471 134
shot1_2 8708 261 10744 322 shot1_2_RKF.jpg 14180 425
shot1_3 19452 583 3670 110 shot1_3_RKF.jpg 21254 637
shot1_4 23123 693 2736 82 shot1_4_RKF.jpg 24524 735
shot1_5 25859 775 2202 66 shot1_5_RKF.jpg 26926 807
shot1_6 28061 841 2135 64 shot1_6_RKF.jpg 29129 873
shot1_7 30196 905 6906 207 shot1_7_RKF.jpg 33500 1004
shot1_8 37103 1112 7007 210 shot1_8_RKF.jpg 40640 1218
shot1_9 44110 1322 6639 199 shot1_9_RKF.jpg 47313 1418
```

⇒ The annotations included the shot name and the annotations of the objects (their names and/or situations e.g. Female_News_Person Standing). The data we received was in the following format:

```
134 Female_News_Person Standing
425 Female_Speech Female_News_Person News_Subject_Monologue
637 Non-Studio_Setting Female_Speech Person Man_Made_Object Monitor Text
735 Female_Speech Graphics
807 Female_Speech Graphics
873 Sound Female_Speech Man_Made_Object
1004 Non-Studio_Setting Sound Male_Speech Person Man_Made_Object Sitting
1218 Sky Sound Female_Speech Man_Made_Object
1418 Non-Studio_Setting Male_Speech Male_Face Microphone Monologue
```

⇒ The speech transcript included the shot name, the confidence of the text's accuracy and the text that is spoken during the shot. Limsi speech recognizer produces this speech transcript by recognizing the audio of these videos. The data we received was in the following format:

```
shot1_1 0.547000 THERE'S
shot1_2 0.896521 BEEN A BALLOON FOSSETT IS NOW OVER THE BLACK SEA IT BUT
WINDS AREN'T COOPERATING AND NEITHER IS HIS EQUIPMENT A. B. C.'S SHEILA
MACVICAR HAS THE LATEST
shot1_3 0.868859 NOT VISIBLE TODAY HIDDEN IN THE FOG ONLY STEVE
shot1_4 0.863184 FOSSETT'S VOICE WAS AUDIBLE AS HE TRIED TO AIR TRAFFIC
shot1_5 0.971628 CONTROLLERS IN ROMANIA THIS MORNING
shot1_6 0.699013 PROBLEMS WERE APPARENT THIS IN
shot1_7 0.538196 A VERY SLOW FLYING SEVERAL WOMEN US THIRTEEN DOESN'T SE
shot1_8 0.982424 THIRTY KNOTS AND NOW DOWN TO TWENTY FIVE MILES PER HOUR
SLOWER THAN WHAT HIS MISSION CONTROLLERS HAD HOPED FOR
shot1_9 0.875723 THE WINDS IN THE VICINITY OF THE BLACK SEA WHERE STEVE
FLYING WERE ABOUT THE SLOWEST ON THE GLOBE
```

3. System Overview:

As we have mentioned before in Introduction section, we integrated different modalities together such as visual information -the video itself-, audio –the speech- and text –the annotations-. We have not only integrated the different modalities together but also moved onto the semantics to make more accurate, efficient and effective searches on the data archive we had. We provide the user much functionality which can be seen in the figure 2. We allow the user to search in the whole data and also we provide some filters such as commercial, sports or face to search in a more specific part of the archive.

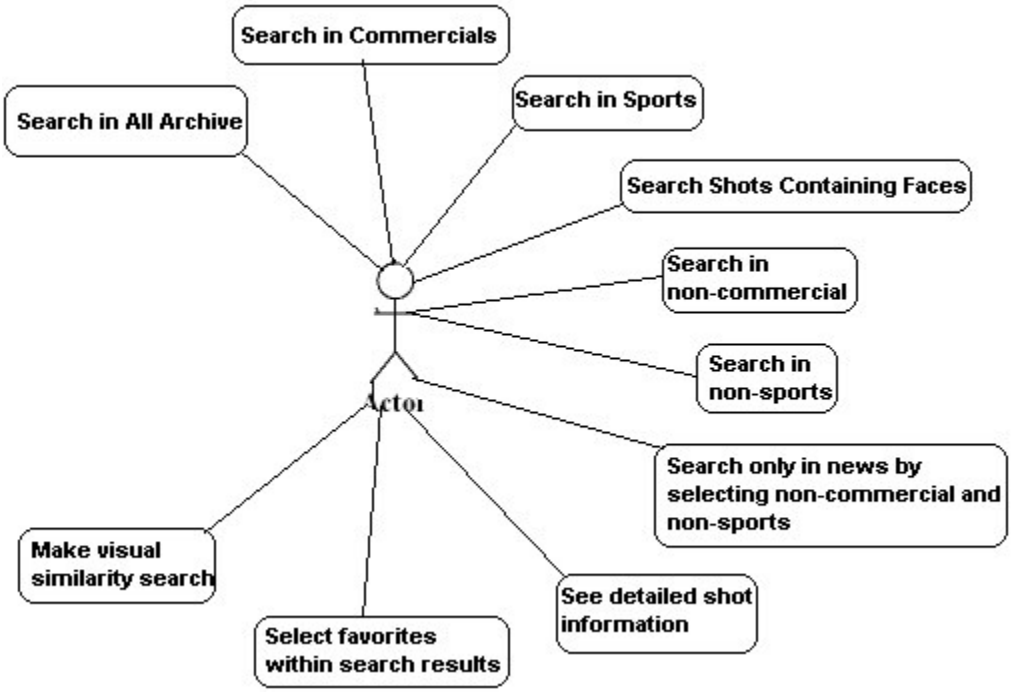


Figure 2: Use Cases

3.1. Low Level:

Before explaining what sub-problems we dealt with to accomplish our goal, let us talk about what we needed and used in low level. Figure 3 demonstrates the sub-modules we implemented in low level. Since there is a non-negligible amount of development in the area of video analysis and understanding we tried to find as much ready to use modules as we could to gain more time on focusing on our main task and goal. Therefore we have found many ready-to use modules in text extraction. Also in low level visual feature extraction we have used java advanced imaging libraries.

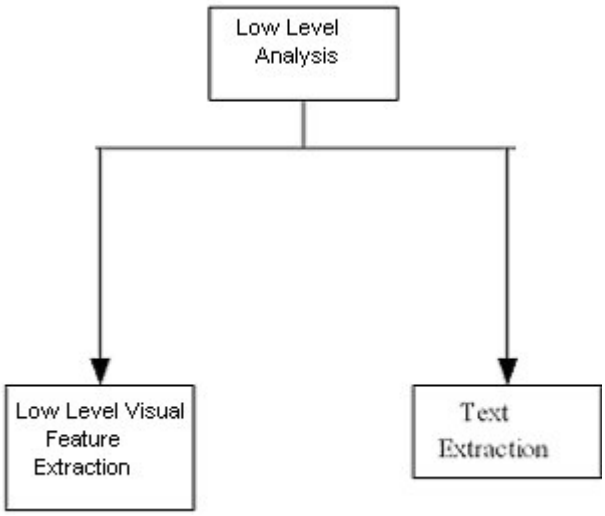


Figure: 3: Low-Level Modules

3.1.1. Text Extraction:

Text extraction and text analysis are important features of our video retrieval system because it is crucial to have efficient and correct results from our database and transcripts. In text extraction we had 4 sub-modules to reach this goal of retrieving efficient and correct results. Figure 4 shows the sub modules of text extraction which are stop word elimination, stemming, tagging and word hierarchy.

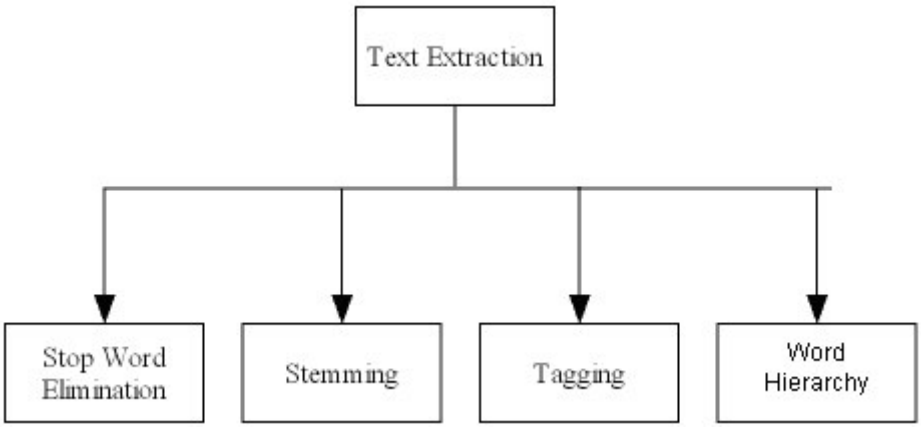


Figure 4: Text Extraction Modules

Initially we had two main transcript files to work on. First one is Annotations file which gives us information about the visual contents of the shots and videos. Database reaches and makes connection to Annotations file by using two integers as parameter: VideoID and ShotId, and the information are kept by key-words like Text_Overlay, Land, and Outdoors.

The other text transcript we used is speech transcript which gives us information about the speeches that are included in a shot or video. Again database reaches and makes connection to speech transcript file by using two integers as parameter: VideoID and ShotId. The exact

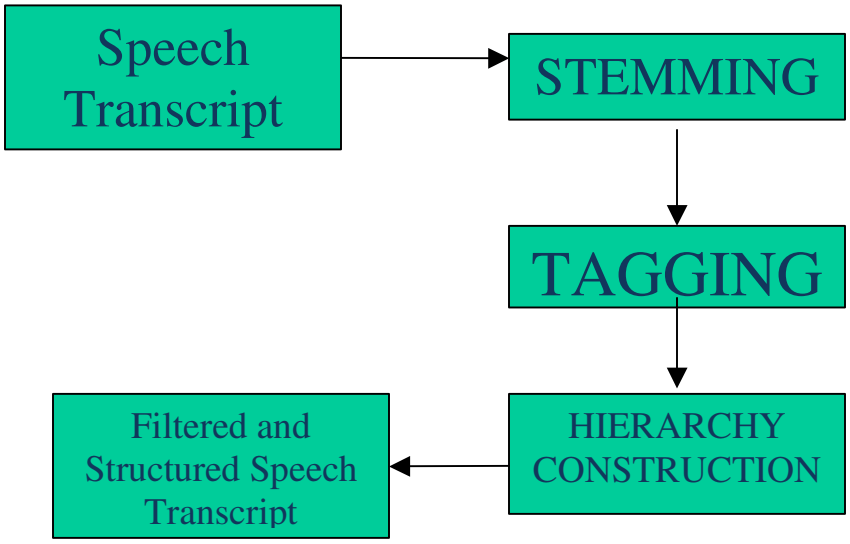
speech of a video is kept as information in speech transcript. In later stages of our project we concentrated on speech transcript.

Why we needed text analysis in our project?

It is very important for us to reach to correct information in most efficient way but we need some analysis and correction on our text sources to reach such a result. For example:

- It is likely to have some typing errors in speech and annotations transcripts (We know that there are many errors in annotations because it is typed by hand by a group of a hundred people but in speech transcript we have a result nearly errorless because it is created by computer based tools). Those errors should be found and corrected before those files are integrated to our system.
- It is very important to find the frequencies of the words because we should work on the words that have a frequency over a limit. The words with a very low frequency are not important for us and they make overhead for system so they should be eliminated.
- On the other hand words like "the, a, is, are" are repeated many times in files but they are not important for us and make an important overhead so they also should be eliminated.

Steps of Text Analysis:



3.1.1.1 Stop Word Elimination:

It is known that the words "the, a, of …etc" are kind of words that do not have a high level meaning but are included in the text most frequently. Additionally there are some words that have a meaning but used just once or twice throughout the whole text but again slow down our queries. We have counted the frequencies of all of the unique words in the speech transcript and we have found out that the most frequent words which are "the, a…etc" and the least frequent words are unnecessary words as some of them are shown in figure 5 on the next page.

ID	Word	Frequei
22596	the	60161
838	and	28064
22854	to	26993
15421	of	22149
1	a	21320
11122	in	18033
22590	that	13966
8786	for	10866
11789	is	9519
15496	on	8389
25082	you	6688
24821	with	6586
22675	this	6169
15501	one	5943
11819	it	5734
1876	be	4931
24420	was	4868
1120	are	4678
1237	as	4546
3049	but	4513
9065	from	4461
1349	at	4195
10234	have	4132
10915	i	3861
11822	it's	3714
14770	n.	3624

ID	Word	Frequei
14770	n.	3624
14541	more	3566
48	about	3502
609	all	3417
10265	he	3416
3092	c.	3390
10196	has	3289
15016	news	3254
24614	what	3196
3075	by	3128
3228	can	3063
23437	two	3037
22864	today	3031
3	a.	2938
22593	that's	2860
14999	new	2850
22640	they	2802
24726	will	2802
25094	your	2785
24627	when	2732
23841	up	2725
15216	not	2697
15254	now	2670
24481	we	2622
20758	so	2596
802	an	2551

ID	Word	Frequei
802	an	2551
10842	hundred	2525
1556	b.	2470
10539	his	2462
12132	just	2456
22602	their	2433
16312	people	2388
8562	five	2278
22712	three	2248
20848	some	2233
9390	get	2223
1948	been	2127
15692	out	2095
15587	or	1909
24673	who	1846
22809	time	1845
22582	than	1831
15127	no	1795
12966	like	1778
7239	eight	1773
10964	if	1771
25036	year	1769
8918	four	1768
23422	twenty	1745
19531	say	1744
15772	over	1657

Figure 5: First 75 words of stop words out of 250.

3.1.1.2 Stemming

It is used to eliminate the morphological and fixed suffixes of the words in transcripts, and to find the roots of the words roots in order to get the semantics correctly and to make our searches easy. Additionally we needed them in order to form a hierarchy of semantics. It is used for only English words. It cannot be used for eliminating prefixes or non-English words. It is a part of normalization process of the words.

Those are some examples of exact working stemmer:

Cooperating	Coop
Equipment	Equip
Tried	Try
Controllers	Control
Absorbed	Absorb
Balloonist	Balloon

We are using Porter's Stemmer [16] for stemming process. It is used for 20 years and originally coded in BCPL but it is coded in many programming languages in a way that is exactly equivalent to the first version coded in BCPL. We are using the stemmer that is coded in Java.

In stemmer there are six steps of process. In step1 –ed, -ing and plural suffixes are eliminated:
After step1 words become as following:

caresses	→	caress
ponies	→	poni
ties	→	ti
caress	→	caress
cats	→	cat
feed	→	feed
agreed	→	agree
disabled	→	disable
matting	→	mat
mating	→	mate
meeting	→	meet
milling	→	mill
messing	→	mess
meeting	→	meet

As you can see there are some false words after step1. Most of them will be corrected in following steps but some will remain false. In step2 "i"s at the end of word will be converted to "y" so words like "poni" will be corrected to "pony". In step3 consecutive suffices are reduced to single suffix. Some examples:

ational – ate
tional-tion
enci-ence
anci-ance
bli-ble
alli-al
entli-ent
iveness-ive
ousness-ous

In step4 suffices like -icate, -ful, -ness, -ize, -ative are eliminated.

In step5 suffices like :-ize, -ive, -ous, -iti, -ate, -ism, -ment, -ant, ence, -ic, -able are eliminated and finally in step6 "e"s at the end of words are eliminated in needed conditions

In later stages of the project we modified the stemmer in a way that it eliminates only the plural suffices so it processes only step1 and step2 but in needed situations other step could be include in stem process.

3.1.1.3 Tagging

Brill's Tagger [2] is used to specify the type of words according to grammatical structures and to classify them in the scope of their own types. It is written by Eric Brill and coded in C. It can be used in Linux/Unix environment.

The output of tagger input "*Ali hold the ball. run Ayse run . Run Ayse run. Fatma look at the horse*" is:

Ali**/NNP** hold**/VB** the**/DT** ball**/NN** **./.** run**/VB** Ayse**/NNP** run**/VB** **./.** Run**/NNP** Ayse**/NNP** run**/VB** **./.** Fatma**/NNP** look**/VB** at**/IN** the**/DT** horse**/NN**

Here the type of word is stated with keys like VB NNP NN. For example VB means verb NNP and NN means noun, but different kinds of noun.

Tagging is done in two stages. Every word is assigned its most likely tag in isolation. Each word in the tagged training corpus has a lexical entry consisting of a partially ordered list of tags, indicating the most likely tag for that word, as well as all other tags seen with that word (in no particular order). A list of transformations is provided for determining the most likely tag for words not in the lexicon. Unknown words are first assumed to be nouns (proper nouns if capitalized), and then cues based upon prefixes, suffixes, infixes, and adjacent word co occurrence are used to change the guess of most likely tag. Next, contextual transformations are used to improve accuracy.

We used tagger to get the nouns and verbs of the text so it was possible for us to have a hierarchy of those analyzed words. On the other hand the objects in the sentences are more likely to be queried if somebody is searching for somebody or some organization. In the speakers' speech, which we retrieved as a text, if we become able to identify the objects of each sentence, our search became more filtered and therefore the results came more quickly.

3.1.1.4 Hierarchy Construction

We needed a hierarchy in some cases where we are making a search. This hierarchy was formed in such a way that words are grouped according to their meanings .We needed this hierarchy in order to understand the semantics better. The hierarchy is formed after the outputs of stemming and tagging: right after we have filtered a word into root, and we have defined the type of the word (see 2nd and 3rd sections). If we are to give an example, "oak" is a "tree", "tree" is a "plant. In this module we used the JWordNet to construct our hierarchy.

Wordnet [15] is a database and network of words linked by semantic and lexical relations. First WordNet was developed for English 15 years ago, but recently there are projects for many languages which are finished or still running.

In WordNet nouns, verbs and adjectives are organized according to their synonym sets. By that way it is possible to reach the information about a word quickly and to organize this information effectively. It is the effective organization of lexicographic data. There are paradigmic and syntagmatic relations between the words. WordNet is a result of psycho-lexicologic phenomena so lexicon [5] is the whole of syntagmatic, phonological and lexical pieces.

There are five main categories of words in WordNet network:

- Nouns
- Verbs
- Adjectives
- Adverbs
- Functional Words

The main relations between words in WordNet are below:

- Synonymy
- Hypernymy / Hyponymy
- Antonymy
- Meronymy / Holonymy
- Entailment
- Truponomy

We need WordNet because we need a hierarchy in some cases where we are making a search. This hierarchy was formed in such a way that words will be grouped according to their hypernymy relations .We need this hierarchy in order to understand the semantics better. The hierarchy is formed after the outputs of stemming and tagging: right after we have filtered a word into root, and we have defined the type of the word. If we are to give an example, "oak" is a "tree", "tree" is a "plant. We are using JWordNet, the Java version of WordNet, just to integrate it to our system more easily. This hierarchy was used to provide user flexibility and more alternatives on taking output. For example if there is no result about the keyword that user is looking for, so user will choose a hypernym degree of hierarchy to be displayed and so all entities under that level will be outputted from our database.

3.1.2. Low-Level Visual Feature Extraction and Similarity Ranking:

While manual image annotations can be used to a certain extent to help image search, the feasibility of such an approach to our large database is questionable. In some cases, such as face of texture patterns [12], simple textual descriptions can be ambiguous and often inadequate for database search. Therefore we used some of the various low-level features, the most appropriates being color and edge features [14].

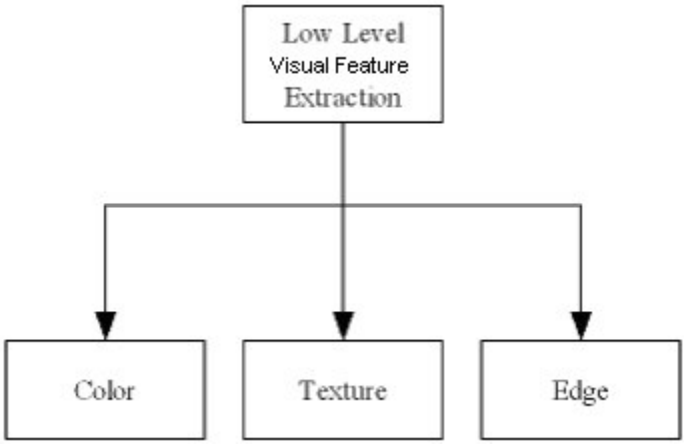


Figure 6: Low Level Modules

We have classified Low Level Feature extraction under three sub modules as shown in figure 6. When processing color we used histogram techniques on the global frames as well as generic partitions and regions of interest of the frames. This method yields successful results in

scenes where there is a dominant color, such as soccer game where there is a green field, weather news when there is a blue background, hockey games when there is the white ice rink and so on. We also computed statistics for the global frames and grids. This was useful because it enabled us to make fast and effective comparisons while keeping an optimum amount of data in our database. The dominating color in commercials, for example, usually has a higher mean and therefore statistics comparisons yield satisfactory results when we try to decide if a scene is from a commercial. Due to the fact that color histogram discards information about an object's shape, location, size, and orientation, we further analyzed the texture [11] information from the frames. We used Gabor[3] wavelets to process texture information and Canny Edge Detector [13] to evaluate boundary information.

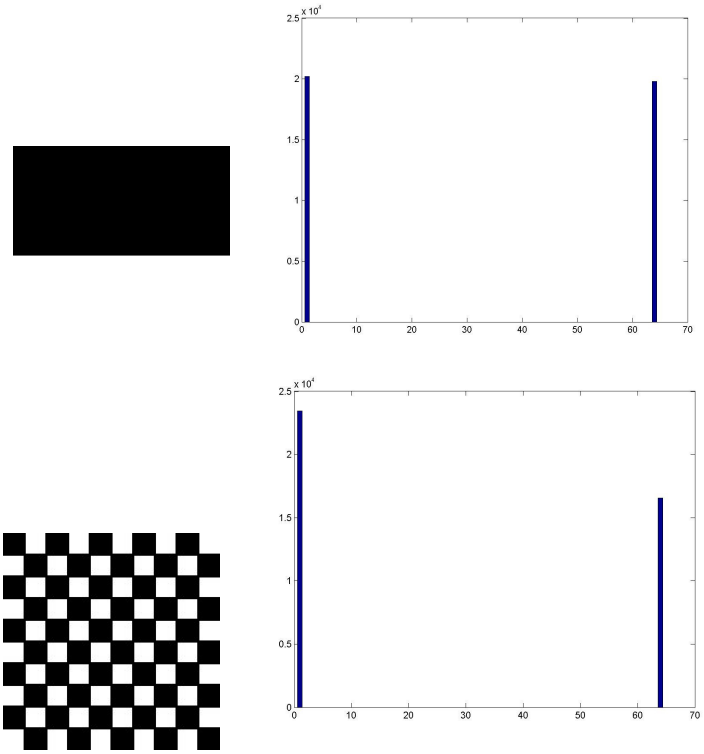


Figure 7. A good example of one of the drawbacks of a global histogram. Due to lack of information about the texture, two totally different images may have similar color histograms.

Histogram based color search characterizes an image by its color histogram but the drawback of a global histogram representation is that information about object location, shape, and texture is discarded as seen in figure 7. Images retrieved by using the global color histogram may not be semantically related even though they share similar color distribution in some results. The color histogram for an image is constructed by counting the number of pixels of each color. We use the RGB color space and to make sure that distinct colors are not assigned to the same bin we represented the images as three dimensional arrays where each dimension corresponds to one of the three basic colors. Suppose the image is w by h pixels. Then the image is represented as a one dimensional array of size w*h. From each pixel we extract three values, first one denoting red, the second green, and the third blue. The large size of the histogram array would impact the performance of database retrieval. Therefore we partition the color values into n bins. For example, when n=4, the color values between 0-63 will be in the first bin, 64-127 in the

second bin, 128-191 in the third and 192-255 in the fourth. Now we have three dimensions like that corresponding to the three colors so we have a 4 by 4 by 4 cubes, which is our histogram. The figure 8 is shown below as an example.

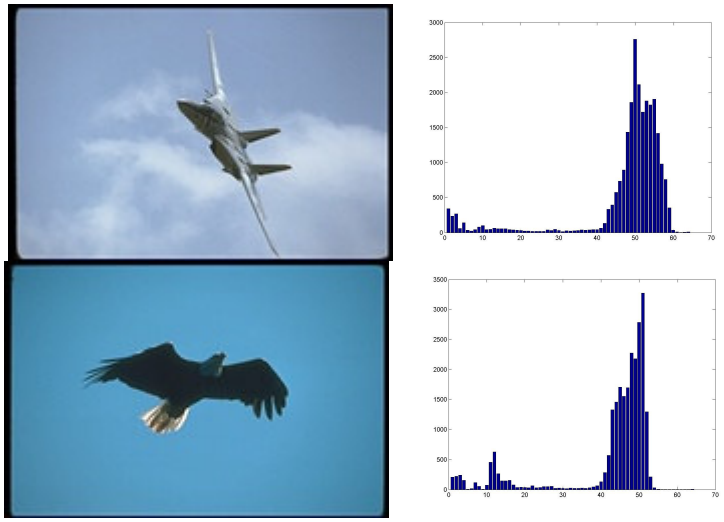


Figure 8. Global histograms of two different images. These two images would be classified to be of the same sort if only global histogram information were to be used.

While scanning through our image array, for each pixel we check the RGB values and then increment the corresponding bin in our histogram. For example, if a pixel has RGB values of (30, 40, 50) since all the values are between 0-63, the $(0, 0, 0)^{th}$ bin of our histogram will be incremented by 1. When the whole image is scanned, we convert the 4 by 4 by 4 histogram array into a one dimensional array of length 4*4*4=64. Then a generic distance comparison is sufficient between two histogram arrays to rank their similarity. A distance comparison is given in figure 9.

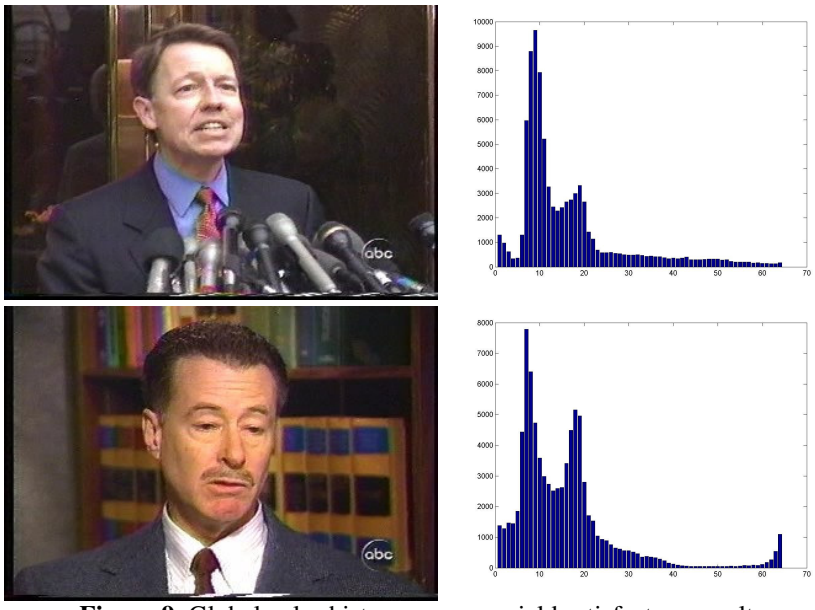


Figure 9. Global color histograms may yield satisfactory results.

Sometimes we are not interested in the global image. Therefore we may specify a region of interest and then take its color histogram. By default we divide the images into 3 by 3 grids and compute separate color histograms for each grid. The grid size is generic and can be adjusted to any sensible number. However, the size of the database multiplies by the number of grids and this affects the performance. Another method we use to keep our results more compact is to calculate statistics for the global and grid images. By calculating the mean and standard deviation of the color values, we are able to make fast and effective similarity tests while keeping a minimal amount of data in our database. The gridded view and histogram example is given below in figure 10.

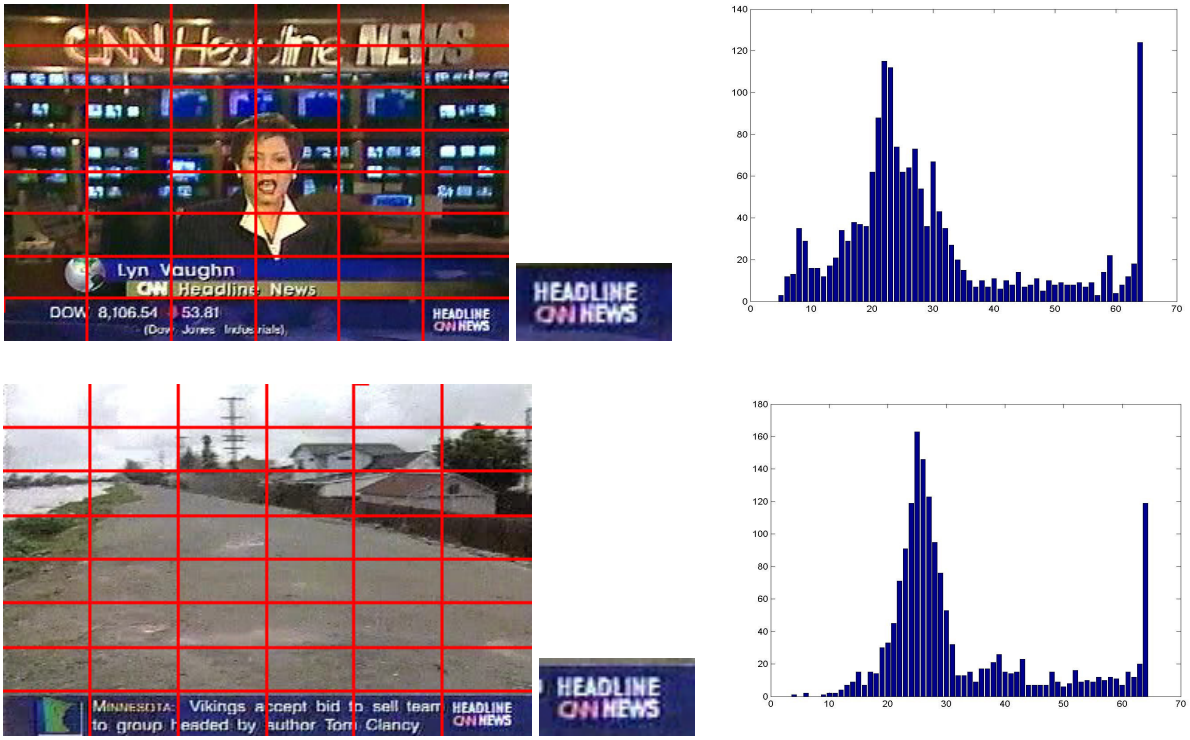


Figure 10. When we partition the images into 8 by 6 grids, the most bottom right grid is helpful in detecting headline news. The variations in the histogram are mainly due to difference of resolution between the different shots.

Texture is a key component of human visual perception. Like color, this makes it an essential feature to consider when querying image databases. Unlike color, texture occurs over a region rather than at a point. Texture has qualities such as periodicity and scale. It can be described in terms of direction, coarseness, contrast and so on. Because of this rich definition, texture is a particularly interesting feature of images and can be extracted in various ways. Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.

Based on these criteria, we use the canny edge detector to obtain edge information for our data set. Canny edge detector first smoothes the image to eliminate the noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixels that are not at the maximum (non-maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non-edge). If the magnitude is above the high threshold, it is made an edge. If the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above the second threshold.

The effect of the Canny Edge Detector is determined by three parameters: the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds used by the tracker. Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The localization error in the detected edges also increases slightly as the Gaussian width is increased. Usually, the upper tracking threshold can be set quite high and the lower threshold quite low for good results. Setting the lower threshold too high will cause noisy edges to break up. Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output. Figure 11 demonstrates the canny edge detector outputs.

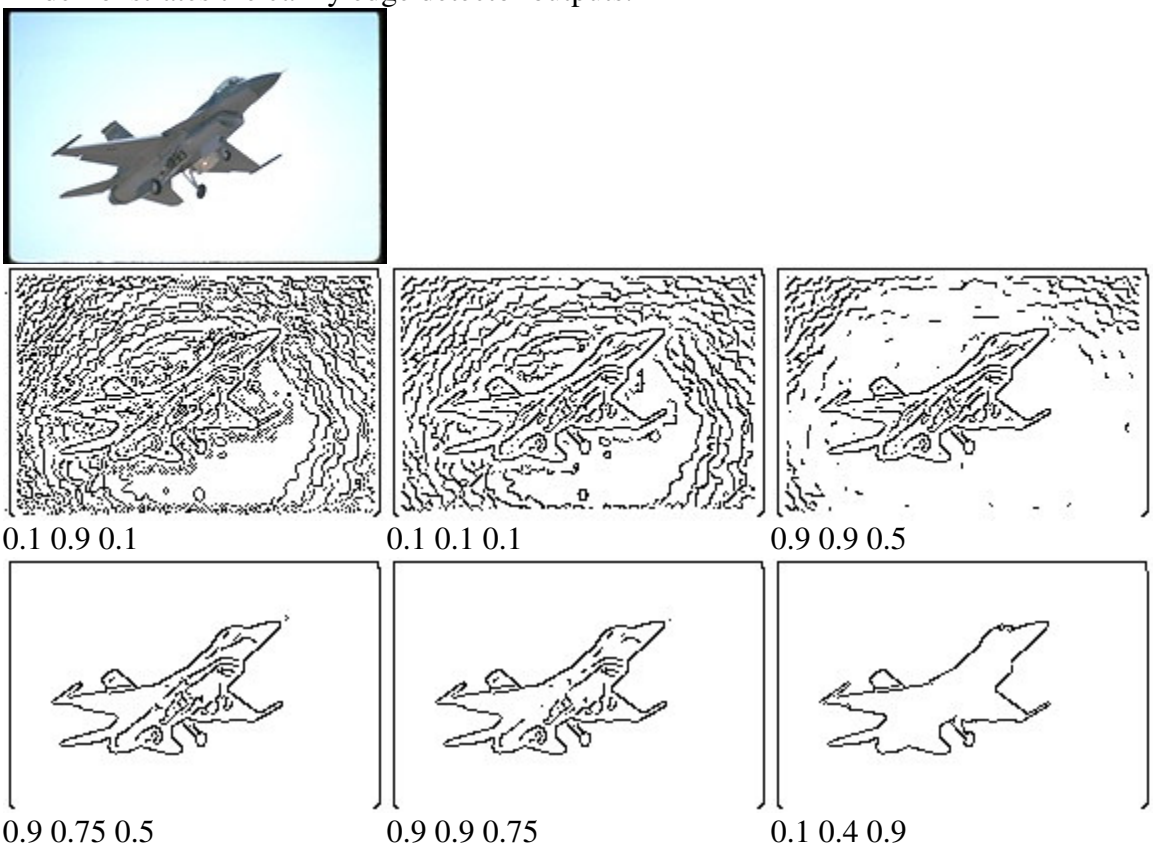


Figure 11. Various outputs from the Canny Edge Detector. The first parameter is the standard deviation of the Gaussian blur kernel, the second parameter is the fraction of the high edge strength threshold, and the third is the fraction of the distribution of non-zero edge strengths for hysteresis. Each pixel in the above images has a corresponding orientation value between 0-2pi radians. By trial and error we find the most feasible parameters (for this image 0.1 0.1 0.9 etc) and then we take the orientation output with those parameters and apply global and grid histogram techniques.

We used the edge orientation output of the Canny Edge Detector. In this output all the pixels that have been detected as edges have an orientation value between 0-2π. By masking out the non-edge pixels, the data size is significantly reduced. We compute a 64 bin global histogram for each image and also for the 3 by 3 grids. Again the number of bins in the histogram and the number of grids are generic and can be adjusted to any sensible number. We also calculate the mean and standard deviation values, as well as the minimum and maximum values. It is then a straight forward distance computation between the statistics and histogram arrays to determine the similarity between any two given images.

When ranking the images by similarity, we compare the Euclidean distance between the image arrays, whether they are color or orientation histograms or statistics outputs. Once an image is selected, other images are ranked in ascending order using pair-wise distances and a similarity list is obtained.

3.2. Mid-Level Analysis:

In the mid-level, we used face detector outputs that was provided to us and the features we have obtained in low-level feature extraction part. Therefore for now we have 2 sub-modules in mid-level which are face detection and scene classification as can be seen from the figure 12. For the face detection, the output of the face detector was the faces in the shots and for the scene classification the features we had were global RGB histogram, 3 by 3 RGB histogram, global RGB statistics (minimal value, maximal value, mean value and standard deviation values) and lastly 3 by 3 RGB statistics (minimal value, maximal value, mean value and standard deviation values). For short, in mid-level, we used what we had until then in order to be able to detect faces, separate commercials from regular news and classifying scenes-currently sports news-.

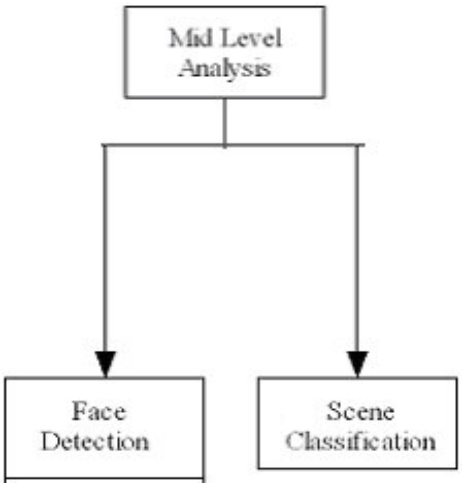


Figure 12: Mid level sub modules

3.2.1. Face Detector:

We used a face detector and it was be a big help for us in queries for people and also in anchor detection for the future. The face detector outputs we use are basically image outputs.

Basically what we did for face detection was to find the key frames among the outputs that had faces in it. The key frames of shots who had faces in it were marked in the database as face positive (1) and the key frames of shots who don't appear in the face list were marked in the database as face negative (0). This can be used when one is searching for a particular person like "Bill Clinton".

We are inserting this information into the database using the method in our *DatabaseEditor* class. For future development we have implemented a *Face* class in order to create face objects and to make operations on them. We have also provided an *insertFace* method that inserts the face information by taking Face objects as input. The detailed information about the classes can be seen in section 6.

3.2.2. Scene Classification:

As we have mentioned before in the Dataset section, our video is composed of many different scenes such as the regular news, sports news, weather news, economy news, and commercials and we needed to detect these stories [9]. The format of the videos is shown below in figure 13 and as you may see the shots that have same scenes come consequently and then followed by some other shots from another scene.

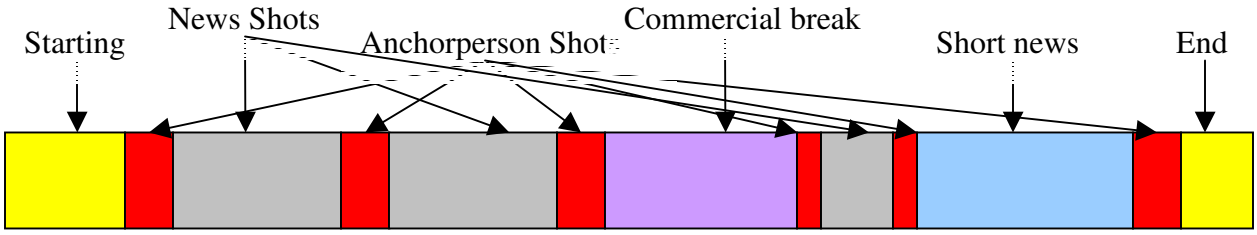


Figure 13: An example shot distribution in a movie

Since we have different scenes, we wanted to classify them in order to have filters in our search. We used 2 classifiers [10] one is textual and the other is visual. We used textual classifiers for sports detection (section 3.2.2.2) and visual for commercial detection (3.2.2.1).

3.2.2.1 Commercial Detector:

Detection and removal of commercials [6] is a very important step for finding real material to work on which is regular news however sometimes one can make search on commercials too. There is no single image feature that allows one to tell the commercials from the news content of a broadcast but a combination of some simple features of the images would work. For now we have used the color features of commercials in order to make our classifier.

All of the videos we have in our dataset were videos however they contained not only the "news" but also the "commercials" that occur frequently. If we became able to exclude the commercials that were unnecessary in our queries then we would have a narrower archive to search for and that definitely would speed up the query performance and the results would become more accurate among news.

As we said before, since the archive we searched for would become smaller, the query results' speed would increase. If we are to give an example, think of offices. If the user is searching for all of the news about offices, and there is also office commercials like "Office Depot" then the video the results will contain the commercials too. However if we filter the video from commercials this information will not be among the query results. Therefore a more accurate and fast query will be performed so what we do is to provide the user to select this filter or not in the graphical user interface as can be seen in section 5.

For commercial detection we have used three different low level features: global RGB histogram, 3 by 3 RGB histogram and 3 by 3 RGB mean. First we have selected commercials as an example set, we have eliminated some non-bright shots, then we have selected news as another example set, we have eliminated too bright shots among this set. The commercial set was our negative set and news set was our positive set. An example from our positive set and negative set are seen below in figure 14 with their color properties:

Figure 14.a: A key frame from Friskies commercial where global RGB means are: 182.874, 182.517, 137.272

Figure 14.b: A key frame from ABC news where global RGB values are: 59.2385, 55.3708, 72.8649

After selecting positive and negative samples from 8 videos, we have used the 3x3 RGB mean values and 3x3 histogram to find the mean of all of the samples' mean and histogram values. After obtaining positive and negative mean out of 3x3 RGB mean values and 3x3 histogram we have took the average of these 2 mean values. We then calculated the distance by Euclidian formula. We took the square root of the sum of squares of the differences between each shot's mean or histogram values and the average of the mean or histogram values.

After calculating the Euclidian distances, we have marked the shots that are nearer to the commercial's mean of the mean values or mean of the histogram values as commercial.

In order to find the mean and histogram values of the positive and negative examples, we have implemented *PosNegClassifier* class, in order to be able to calculate the distances we have implemented *DistanceCalculator* class, in order to be able to mark the nearer shots as commercials we have implemented *CommercialDetector* class. They will be explained in detail in section 6.

3.2.2.2 Sports News Detection

We have implemented the sports version of scene classification module. We are using a textual classifier. There are some special words that are used in sports news such as "soccer, goal, golf, hockey, baseball, basketball, volleyball, football, home run" which are more frequent in sports news obviously. We find the frequency of these words among the speech transcripts of positive and negative examples and take the mean of the frequency of these words. Then we find their frequency distances from the positive and negative samples by again using Euclidian distance. Later we mark the shots that are nearer to sports' mean of frequencies.

After this, we apply some basic properties. If the search is for hockey, we bring shots that contain high values of RGB in visual search part, if search is for soccer; we search for high values of G and so on. An example shot is shown below in figure 15:

Figure 15: The result of hockey search brings this shot with speech transcript:
"THE ICE WHERE WERE THE STREETS AND OFFER U. S. OLYMPIC HOCKEY COACH RON WILSON IS WASHINGTON CAMPS LOSS OF A LIKING". The RGB mean values of this shot are: 198.809, 198.92, and 198.575

4. Database

We have put almost all of the data we have preprocessed in order to have quick access to the features of the shots in each video. The database diagram is given in figure 16. We have the following tables in our database:

AnnotationKeys
Annotations
CommercialDistance
Faces
GlobalHistogram
GridHistogram
GridMean
HistogramDistance
MeanValues
MovingObjects
Shots
SpeechKeys
SpeechTranscript
StdValues
StemMap
StemmedSpeech
Video_Summary_Information

AnnotationKeys (Key, Annotation):
We have assigned a key to each unique annotation word and this table contains all of the unique annotation words with their keys.

Annotations (VideoID, ShotID, Annotation):
We have put the annotations of the shots in this table. VideoID represents which video it is and ShotID represents which shot in that video it is. Annotation contains all of the pre-determined annotation words for that shot.

CommercialDistance(VideoID, ShotID, DistancePos, DistanceNeg):
This table contains for each and every shot the calculated Euclidian distance from positive and negative means of the RGB grid mean values. VideoID represents which video it is and ShotID represents which shot in that video it is. DistancePos is the distance from news videos' mean of means, DistanceNeg is the distance from commercial videos' mean of means.

Faces (VideoID, ShotID, FaceID, Xcoord, Ycoord, Width, Length, Confidence):
This table is prepared for future use and will contain all information about a face output found in the key frame of a shot. VideoID represents which video it is and ShotID represents which shot in that video it is. FaceID is added automatically for each face in a key frame, Xcoord and Ycoord is the upper left corner of the face detected, width and length is the bounding rectangle's width and height, confidence is the probability that it is a face.

GlobalHistogram (VideoID, ShotID, Histogram):

This contains the histogram values of the key frame as a whole. VideoID represents which video it is and ShotID represents which shot in that video it is. Histogram is a 64 bin histogram containing the values.

GridHistogram (VideoID, ShotID, Histogram):
This contains the histogram values of the key frame as a 3 by 3 grid image. VideoID represents which video it is and ShotID represents which shot in that video it is. Histogram is a 9x64 bin histogram containing the values of the 9 grids.

GridMean (VideoID, ShotID, Mean):
This contains the mean values of the key frame as a 3 by 3 grid image. VideoID represents which video it is, and ShotID represents which shot in that video it is. Mean is a 9x3 string array containing the values of the 9 grids' RGB mean values.

HistogramDistance (VideoID, ShotID, HistogramID, PosDistance, NegDistance):
We have calculated the positive and negative histogram distance for each 9 grids of a key frame and put the information such as VideoID represents which video it is, and ShotID represents which shot in that video it is and HistogramID is the grid number from 1-9. PosDistance and NegDistance is the key frame's distance between itself and the news' videos' mean of grid histogram values and the key frame's distance between itself and the news' videos' mean of grid histogram values.

MeanValues (VideoID, ShotID, R, G, B):
This contains the mean values of the key frame as a whole image. VideoID represents which video it is, and ShotID represents which shot in that video it is. R is the mean of Red values in the image, G is the mean of Green values in the image and B is the mean of Blue values in the image.

MovingObjects (VideoID, ShotID, MovingObjectID, Xcoord, Ycoord, Width, Length, DirectionAngle, Speed):
This table contains all the information about a moving object in a shot. This is prepared for future use. VideoID represents which video it is and ShotID represents which shot in that video it is. MovingObjectID is given automatically for each moving object inputted to the database. Xcoord and Ycoord are the coordinates of upper left corner. Width and length is the bounding rectangle's width and height of the moving object.

Shots (VideoID, ShotID, Time_Start, Frame_Start, Time_Duration, Frame_Duration, Keyframe_Name, Keyframe_Time, Keyframe_Frame)
This contains all the information for each shot. VideoID represents which video it is, and ShotID represents which shot in that video it is. Time_Start is the start time of the shot in milliseconds, Frame_Start is the starting frame number, Time_Duration is the duration of the shot in milliseconds, Frame_Duration is the number of frames during the shot, Keyframe_Name is the name of the key frame file of the shot, Keyframe_Time is the point of time where key frame appears and Keyframe_Frame is the number of the frame where key frame appears.

SpeechKeys (ID, Word, Frequency)
This table contains each of the unique word that appears in the speech transcript where ID is incremented automatically. Word is the unique word in the speech transcript; Frequency is the number of times the word has occurred. This table is present in order to detect stop words and eliminate them.

SpeechTranscript (VideoID, ShotID, Transcript)
This table contains the speech transcript of each shot. VideoID represents which video it is, and ShotID represents which shot in that video it is. Transcript is the speech transcript of the shot.

StdValues (VideoID, ShotID, R, G, B):
This contains the standard deviation values of the key frame as a whole image. VideoID represents which video it is, and ShotID represents which shot in that video it is. R is the standard deviation of Red values in the image, G is the standard deviation of Green values in the image and B is the standard deviation of Blue values in the image.

StemMap (OrigID, StemID, Original, Stemmed):
This table is a map between the stemmed words and the original words in the speech transcript. OrigID is the key of the original word, StemID is the key of the stemmed word, Original is the original word; Stemmed is the stem of the original word.

StemmedSpeech (VideoID, ShotID, Transcript):
This table contains the speech transcript of each shot with the stemmed versions of the words in the original speech transcript. VideoID represents which video it is, and ShotID represents which shot in that video it is. Transcript is the stemmed speech transcript of the shot.

Video_Summary_Information (VideoID, ShotID, Anchor, Graphics, Logo, SceneType, StoryType, Commercial, Face):
This table is general information about the shots in the videos. It is partially used currently and there are fields to be filled eventually. Anchor, Graphics, Logo, StoryType is reserved for future use. SceneType represents for now whether it is sports news or not. Commercial represents whether it is a commercial or not. Face represents if there is face in that shot's key frame.

Database Diagram

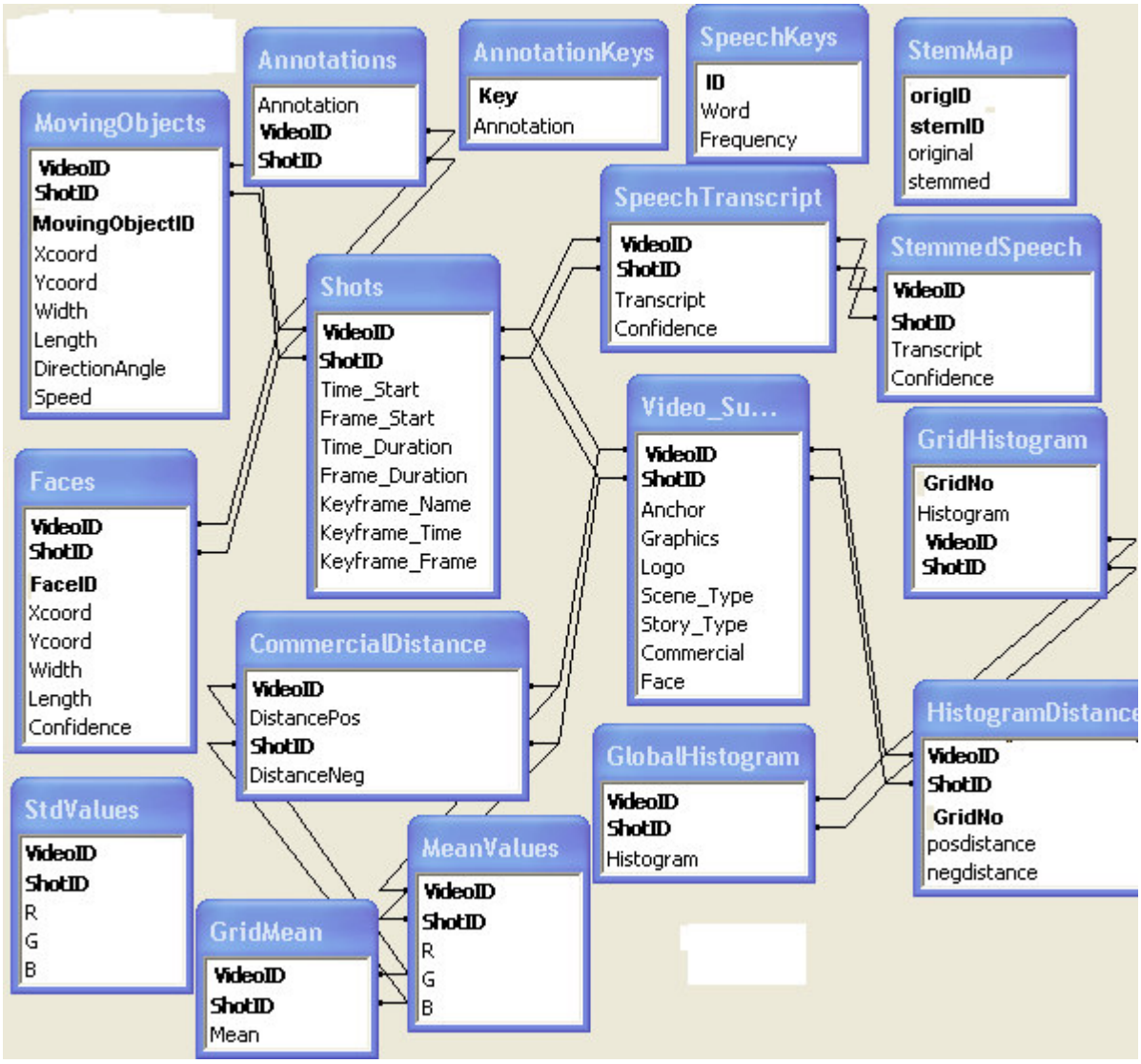


Figure 16: Database relationship diagram.

5. Graphical User Interface

Our user interface is formed of 3 parts. Top panel, middle panel and the bottom panel. The top panel includes components for the search operation, the middle panel includes components for displaying the detailed results of the search operation and the bottom panel includes components about the favorites part.

SCENARIOS

Scenario 1: Text Search with the key "Office"

1) First we enter "offices" to the searchField with the default-neutral- values of commercial, sport and face.(no filtering)

2) When we click the searchButton first the "offices" go to the stemming process and stem process returns with the value office. Then a database connection is established. And the SpeechTranscript table is searched for the entries including office. After search is completed and all results are found -max.20- results are shown in the first page. Moreover, information about the page is given in the resultLabel below. Can be seen in figure 17.

3) Then when we select any image, the shot properties for the image is given in the right with the order: startTime, endTime, startFrame, endFrame, frameDuration, keyFrameTime, keyFrameName, speechTranscript, annotation. Also a big shot of the image is shown in the right

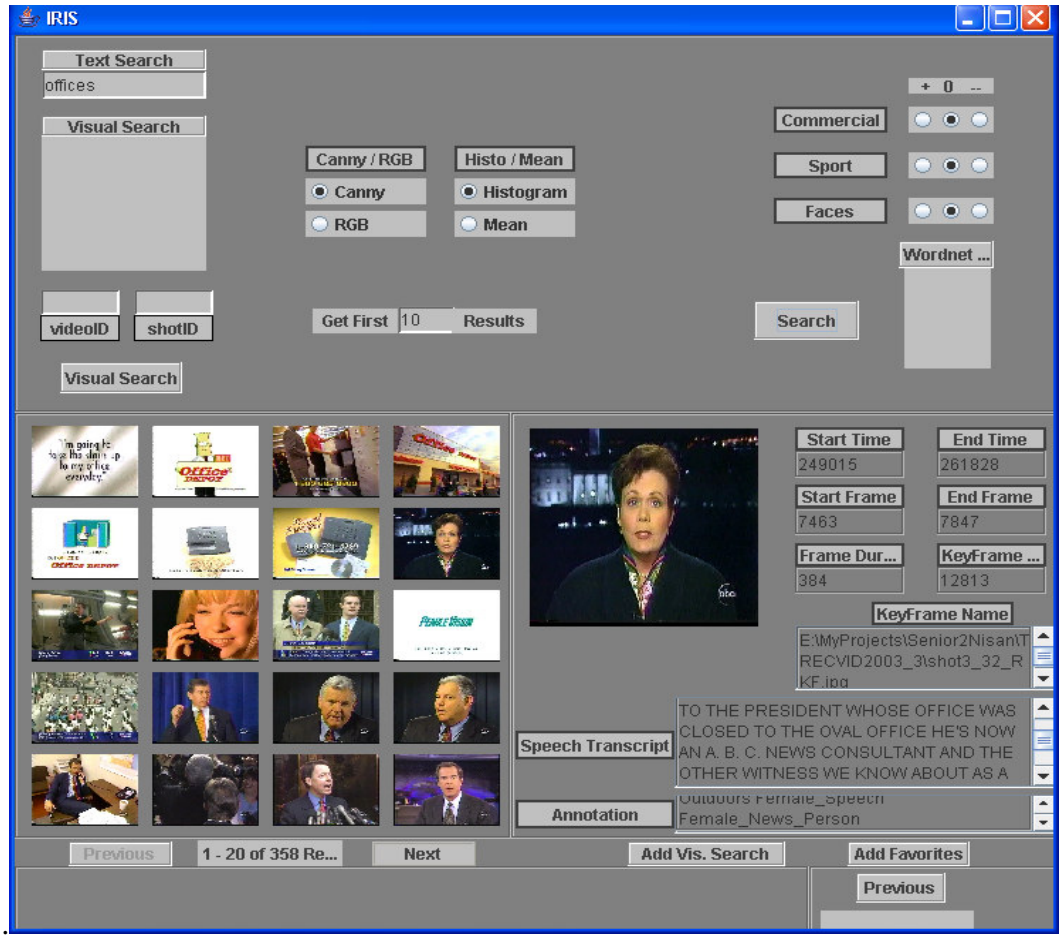


Figure 17: "Offices" search in all speech transcript

4) Then we enter “offices” to the searchField with selecting the commercial option and remaining the face and sport options in default.

5) When we click the searchButton it is seen that we have less results then the previous search because we wanted the results which “must be” in commercials. As seen all results are from commercials. Can be seen in figure 18.

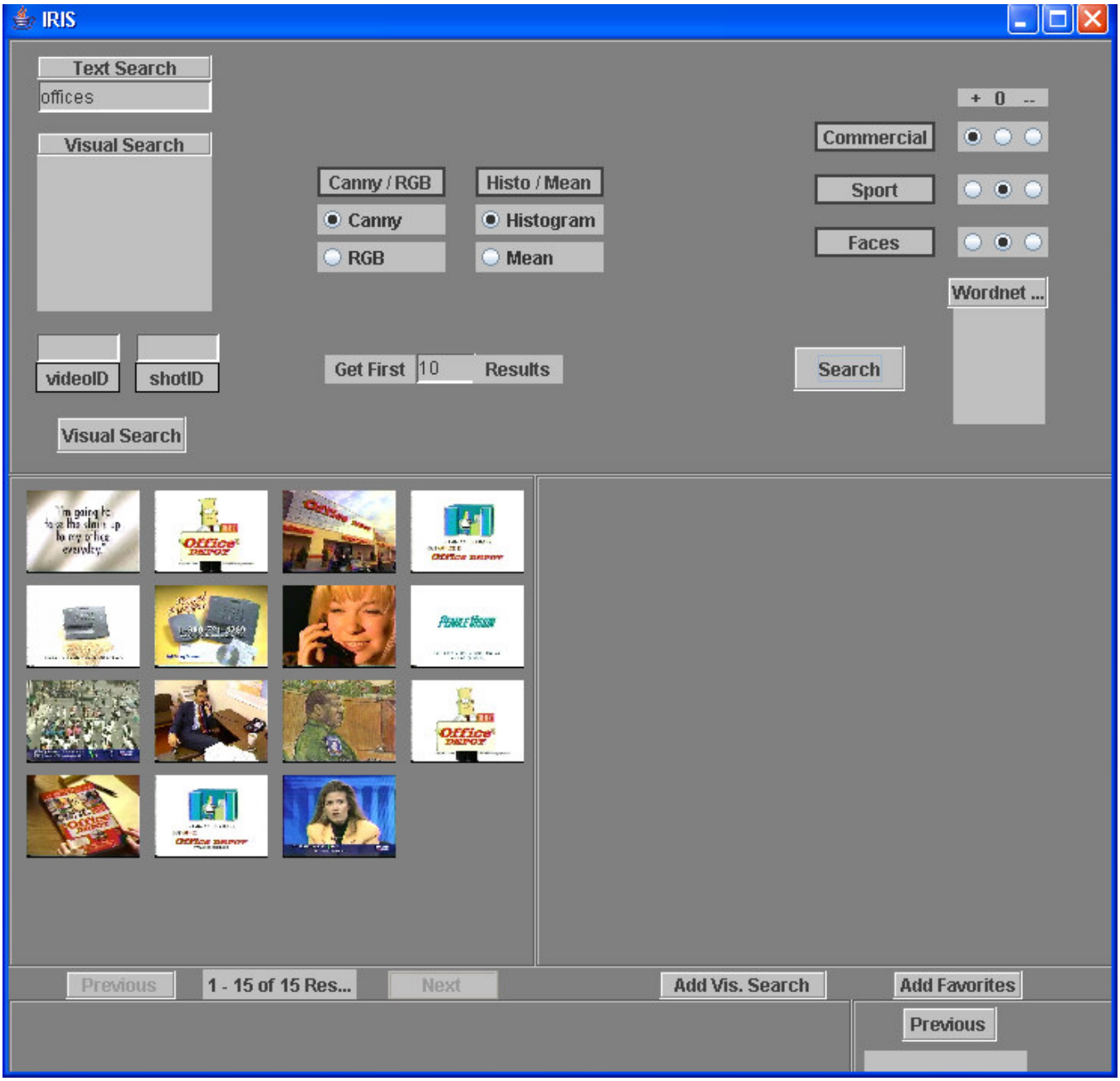


Figure 18: “Offices” searched in commercial shots

Scenario 2) Text Search with the key "Basketball"

1) First we enter "basketball" to the searchField with selecting the sports option and remaining the face and commercial options in default.

2) When we click the searchButton we get the results with the key "basketball" which "must be" in sports. The results are shown in figure 19:

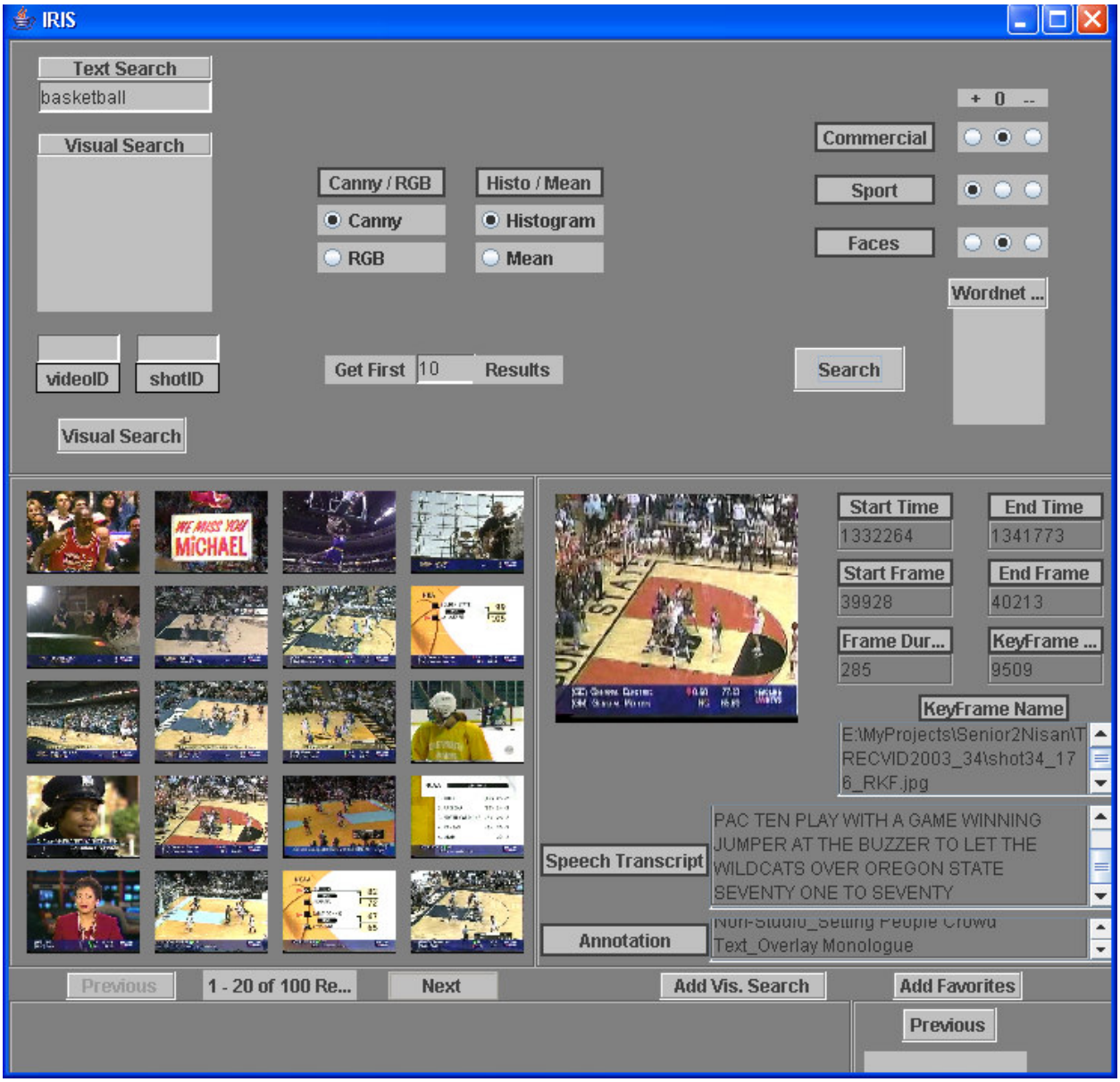


Figure 19: The search results for "basketball" among sports news

Scenario 3) Text Search with the key "Clinton"

1) First we enter "clintons" to the searchField with selecting the face option and remaining the sport and commercial options in default.

2) When we click the searchButton we get the results with the key "clinton" which "must be" in faces. See figure 20,

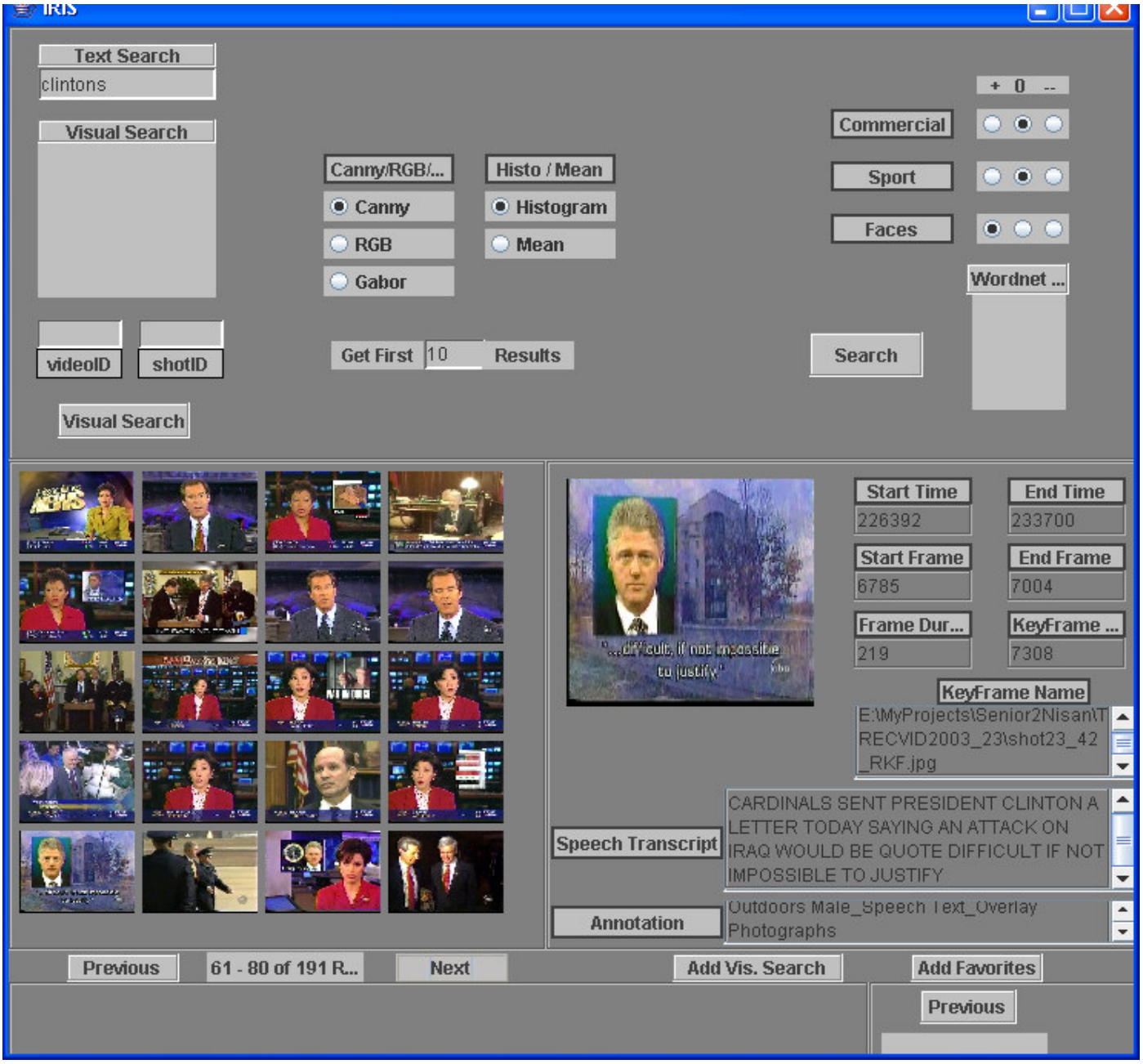


Figure 20: "Clintons" search in containing faces

Scenario 4) Visual Search following the Text Search with the key "Nino"

1) First we enter "nino" to the searchField with the default -neutral- values of commercial, sport and face.

2) When we click the searchButton we get the results with the key "nino" at the mid panel. See figure 21.

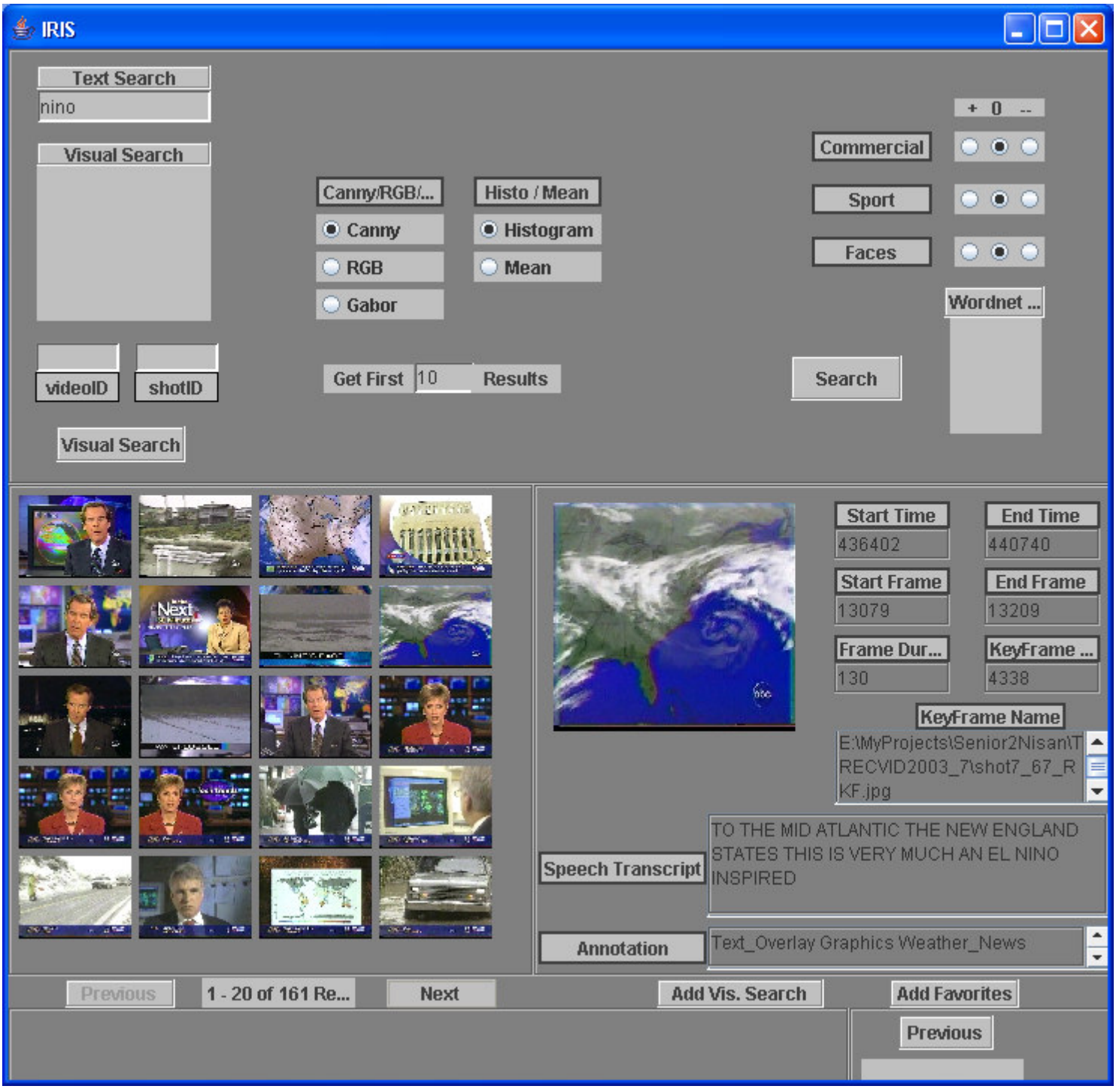


Figure 21: 1-20 results of "nino" in speech transcript

3) Then we clicked the next button and see the results from 21 to 40.
4) Then we select a shot from the list and see the detailed properties of it in the right.
5) Then when we click the Add Favorites button, the shot related to selected image is added to my favorites part as shown. The favorites part can take 10 images once. If the number favorite images exceed 10, user can look the previous/next pages by the previous/next buttons in the favorites part.

6) Then let's add the selected image to the visual search part by clicking the Add to Visual Search button, while the image's properties are shown on the right. See figure 22.

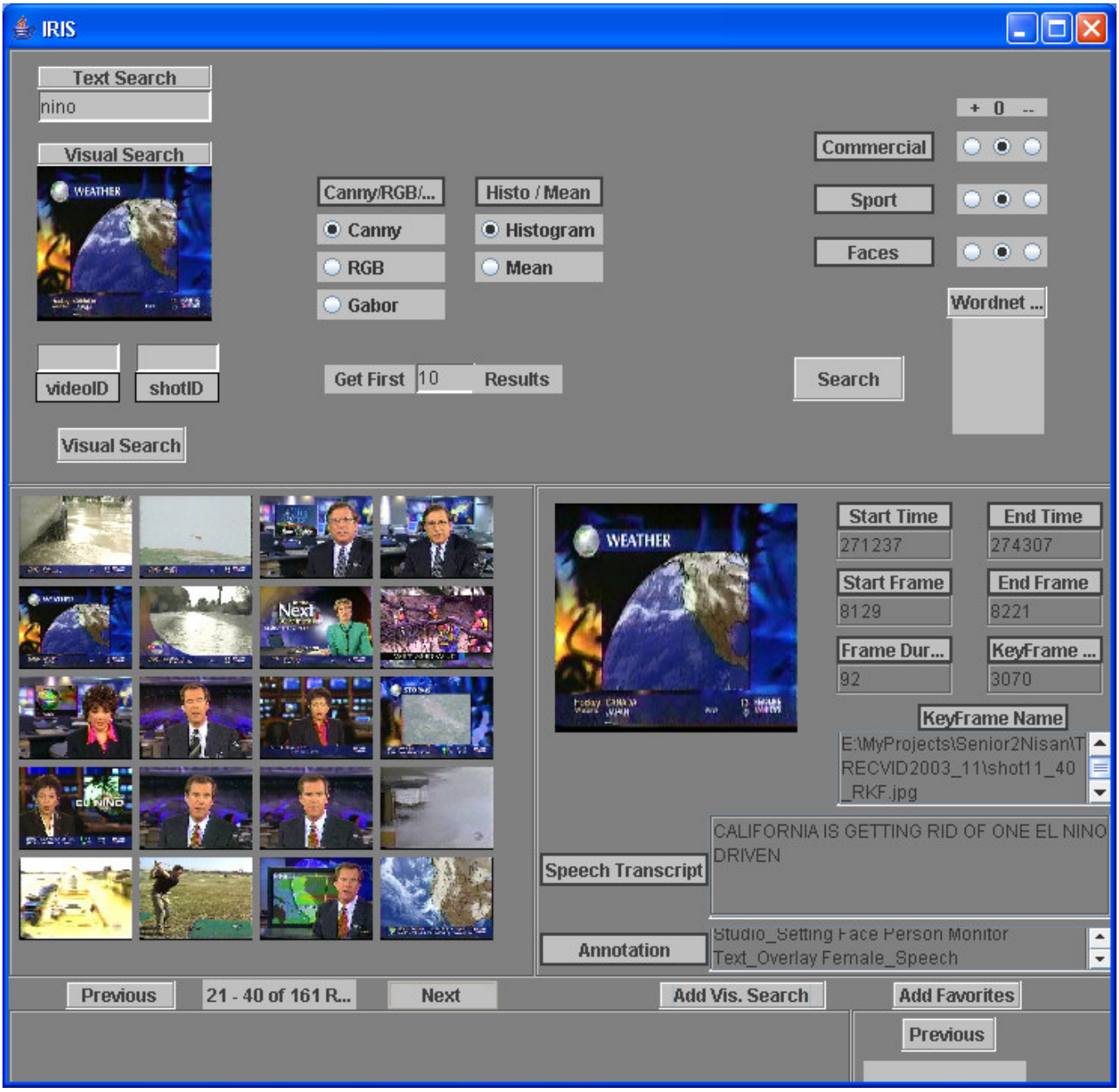


Figure 22: 21-40 results and a result added to the visual search part by clicking on add visual search

7) Now we have an image for visual search. And after selecting RGB&Histogram options and remaining #results field on default value, 10, we click the visual search button to execute visual search of the image by using RGB&Histogram
8) At the end we see 10 images in the result pane. The results are arranged by the method that the most similar image is at first. As seen the first 7 images are so similar to the images searched.

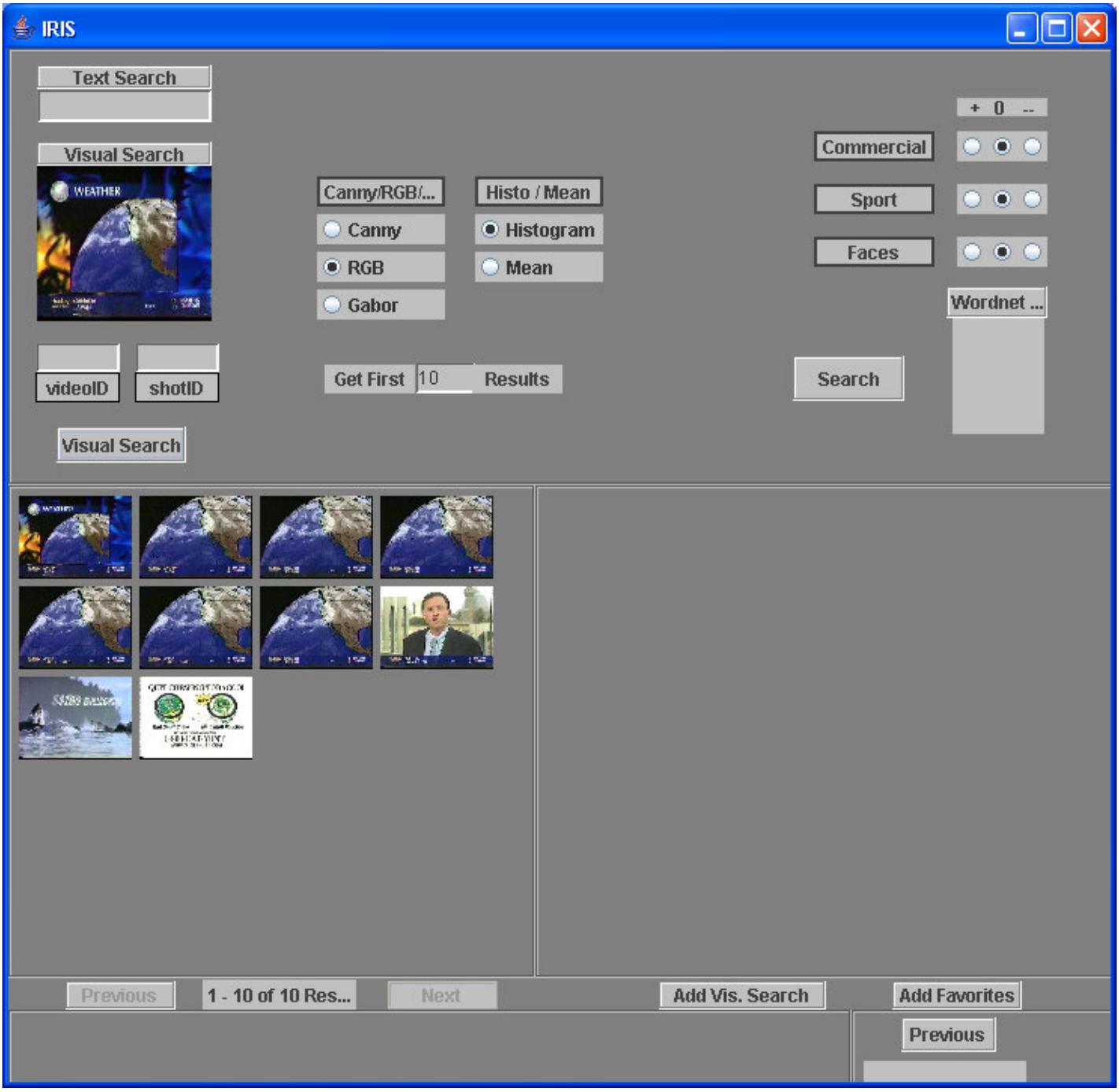


Figure 23: Visual search results…

6. Classes

In this section we are providing the brief descriptions of classes we had in our project.

AnnotationIDs: Finds the ids of the annotations for each of the words in the annotations file and enters the ids of the annotations for each video and each shot in the database in the following format: VideoID ShotID AnnotationID

AnnotationsMerge: Merges all of the *.annotations.txt files into one file that contains the unique list of annotations for each shot.

BoundingRectangle: We have created this rectangle class in order to be able to hold the face and moving objects' properties more organized. Each face and moving object should have xcoord, ycoord, width and length but with this way they just have one attribute which is a BoundingRectangle object.

CommercialDetector: It goes to the CommercialDistance and HistogramDistance tables and marks shots as commercial (puts 1 in the commercial field in the video_summary_information table) whose distance is less far from commercial samples (negative samples).

DatabaseEditor: It deals with all tables in the database and it retrieves information from the database for shots, it inserts information in the database. This information is the preprocessed information that we did. For instance it gets the annotation for the shot, it enters the grid stats and global info.

DatabaseSearch: It queries the database for making search in speech transcript, search in stemmed speech transcript, search in annotation.

DistanceCalculator: This class calculates the distance from negative and positive samples' means for each shot and each grid of each shot. It then puts this information in the database.

Face: This class was made for future use. It has the properties of a face we will find which are xcoord, ycoord, width, height, confidence.

GlobalHistEntry: This puts the histogram information that is read from the files into the database.

MovingObject: This class was made for future use. It has the properties of a moving object we will find which are xcoord, ycoord, width, height, speed, direction angle.

PosNegClassifier: This class retrieves the histogram or statistics values for each shot we have put in our positive or negative example sample set. It then finds the mean of this data-either histogram or mean-.

Shot: This is a shot object with properties: video id, shot id, annotation, speech transcript, key frame name, start time, start frame, time duration, frame duration, end time, end frame, key frame time, key frame number, speech confidence. This object is the object that made almost everything integrable. Between 2 classes these objects were the key objects that were holding the connection.

ShotMerge: This class merged all *.shots.txt files into one shot file that contained all information about a shot and also this class corrected the shot durations by calculating them again since there were many errors in these durations.

SpeechKeyFinder: Finds the ids of the speech transcript for each of the words in the speech transcript file and enters the ids of the speech transcripts for each video and each shot in the database in the following format: VideoID ShotID SpeechID

SpeechStemmapper: This class retrieves the speech transcript of each shot, for each word in this transcript, finds its mapped stem and constructs a new speech transcript that contains only the stemmed versions of the original words in the text. Then inserts this new speech transcript to the stemmed speech table in the database.

StatisticsDBEntry: Enters the statistical information for each shot in the database.(mean, stddev, min, max values).

WordProcessor: In this class the main process in done in a main class by calling LowerCaseConverter, WordListMaker, alphSorter, sortByFreq, StemProcessor, OrigStemMapper respectively.

LowerCaseConverter: This class converts all letters of transcript to lower case. We need this process because tagger gives true output with input of lowercase. As a result of this process the original text is outputted in text file with lowercases.

WordListMaker: The output of LowerCaseConverter enters to WordListMaker. This outputs the list of words in the transcript uniquely and the list words with their frequencies. It then outputs them in two text files.

AlphSorter: This takes the list of words with their frequencies and lists them alphabetically into a text file.

SortByFreq: This takes the list of words with their frequencies and lists them respected to their frequency into a text file.

Stemmer: This is the class where stemming process and its steps are defined.

StemProcessor: This class takes the list of words outputted from WordListMaker as input, and by using the methods of Stemmer class outputs the stemmed versions of those words

in a text file. Moreover, it is used by the GUI to find the stemmed version of the words inputted by user for search.

OrigStemMapper: This class takes the output of StemProcessor and list of words outputted from WordListMaker as inputs, and outputs the map between them.

Tag: This class calls the tagger implicitly and takes the output of it in a text file.

DistNounVerb: This class makes the list of nouns and verbs in two text file by using output of tag.

MapNounOrig: This class makes a map between the list of nouns and the list of words outputted by WordListMaker.

WordTree: This class saves the hypernym hierarchies of words in a text field.

Distance.java: This class reads the color, texture, and edge histograms and statistics data of frames from text files and performs an online pair-wise comparison of the frames and forms a similarity ranking list. The comparison between two frames is computed using Euclidean distance formula.

ImageBuffer.java: This class stores the primitive information about an image. The image originally is in various formats but this class processes the image into an integer array. If the image is in RGB color space, the array has three channels, if the image is the edge output from the canny edge detector, the array has a single channel, if the image is the output from the Gabor filter with say scale:3, orientation:4, the image array has 12 channels etc. This class can also partition image into generic size grids or regions of interest.

FeatureExtractor.java: This class performs statistical operations on the image such as calculating mean, standard deviation, histogram, etc.

ImageProcessor.java: This class performs operations such as detecting edges, storing orientation output, applying Gabor filters. For edge detection and Gabor filtering it uses the Canny Edge Detector and Gabor Filter which are external programs.

7. Summary and Future Work

In this document we have explained what stages we had in our senior design project IRIS. We made video analysis and retrieval software in order to automate the current process of manually entering data or searching for information in video by watching it as a whole –at least by fast forwarding-. The goal of IRIS was to provide people efficient and effective searches on huge video archives in an automated way.

In order to be able to reach our goal, we have designed our project to have 2 sub-modules, one is the low-level feature extraction sub module that dealt with the raw data and produced more specific and structured data and the other one is the mid-level module where more advanced techniques were used and the structured outputs of low-level feature extraction module were interpreted.

We made low level extraction of the data's low level visual features and text. Low level feature extraction provided us the statistics (mean and standard deviation), histogram and edge information about the shots' key frames. We provide these information also for the gridded key frames of the shots. At the same time we have processed the text in low-level which was the speech transcript and annotations files. Transcript was stemmed, tagged and then a word net was constructed. We have then constructed a database in order to put all of these outputs in one place and in an organized way.

After that we have moved onto the second stage which is the mid-level analysis. In mid-level we have used face detector outputs to mark the shots that contain faces in it. This is useful for "people" searches. Then we have made classifications of the scenes. We worked on commercial and sports news' classifications. For commercial we used visual classifiers and for sports we used textual and visual classifiers. After detecting faces, commercials and sports news we then entered these information about the shots of videos in out database we constructed previously.

Then we have prepared a user interface that integrates all we had implemented on a raw dataset which we have selected to be CNN and ABC televisions' news videos in 1998 from TRECVID 2004. Our user interface right now allows visual distance search, textual search, textual and visual search together and lastly wordnet search.

What we did was the low and mid levels of our project. For the high level implementation: story segmentation and learning the correspondence can be done. Story segmentation can be done by using the mid level work such as commercial detection, sports detection and it can be further extended by weather news and economy news detection. Learning the correspondence can be done by aligning the text with the visual part. In order to be able to do these new textual and visual classifiers can be done which is in fact not much different than what we did in sports and commercial detection part. Additional modules like moving object detection and anchor detection can be added as mid-level modules in the future and adding these modules and integrating these in the project was made easy since everything was meant to be related to each other with a Shot object.

Appendix:

Annotation:
Explanatory and defining keywords for shots [7].

Frame:
(Encarta Online Dictionary) single picture on strip of film: any one of the individual pictures that make up a strip of movie film, or a single exposure on a strip of photographic negative or slide images

Image: (lookup: frame)

Key-Frame:
The video is divided into shots and the "key-frame" is a frame that represents almost all of the properties of that particular shot. It is generally selected in random order and in our case it is generally one of the frames that are on the middle of the shot.

Multi-modality:
Using various (more than one) means of filters, features, ways, and classifications for our video retrieval system.

NIST: National Institute of Standards and Technology

Scene:
(Encarta Online Dictionary) the characteristic environment in which an activity or pursuit is carried out in the video

Shot:
(Encarta Online Dictionary) continuous uninterrupted film sequence: a continuous action or image on the screen that appears to be the result of a single uninterrupted operation of the camera It is mainly consisting of similar scenes and images.

TRECVID: Text Retrieval Conference Video Evaluation

Video (Story) segment:
The parts or sections into of the video that is concurrent within itself and mainly consisting of the same subject.

9. References

1. A. Hauptmann et.al., “Informedia at TRECVID 2003:Analyzing and Searching Broadcast News Video", TREC (VIDEO) Conference, 2003.

2. Brill, Eric. “A Simple Rule-Based Part of Speech Tagger”.Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing, (1992).

3. Gabor filters. http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/computerVision/image Processing/wavelets/gabor/gaborFilter.html

4. P. Duygulu and H. Wactlar “Associating video frames with text" Multimedia Information Retrieval Workshop, in conjuction with ACM-SIGIR, 2003.

5. P. Duygulu, K. Barnard, N. de Freitas, D. Forsyth, “Object recognition as machine translation: learning a lexicon for fixed image vocabulary", ECCV 2002.

6. P. Duygulu, M.-Y. Chen, A. Hauptmann, “Comparison and Combination of Two Novel Commercial Detection Methods", ICME 2004.

7. P. Duygulu. Cogulortam Veri Madenciligi :Cok-kipli verinin daha iyi erisim ve degerlendirme amacli tumlesimi”, Bilkent University, 2004.

8. TRECVID 2004, http://www-nlpir.nist.gov/projects/tv2004/tv2004.html

9. T.-S. Chua, Y. Zhao, L. Chaisorn, C.-K. Koh, H. Yang, H. Xu, “TREC 2003 Video Retrieval and Story Segmentation task at NUS PRIS", TREC (VIDEO) Conference, 2003.

10. S. Chang and A. Hsu. Image information systems: Where do we go from here? IEEE Trans. on Knowledge and Data Enginnering, 4(5):431{442, October 1992.

11. , Howarth, Peter. Stefan Ruger. *Evaluation of Texture Features for Content-based Image Retrieval*, Department of Computing, Imperial College London

12. B.S. Manjunath and W.Y. Ma. *Texture features browsing and retrieval of image data*. IEEE Transaction on Pattern Analysis and Machine Intelligence, 8(18):837-842, 1996.

13. J. Canny *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov. 1986.

14. David G. Lowe. Distinctive image features from scale-invariant keypoint, *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110.

15. Fellbaum, Christiane. *WordNet: An Electronic Lexical Database*. MIT Press

16. Porter, Martin. *The Porter’s Stemming Algorithm*. http://www.tartarus.org/~martin/PorterStemmer/