AUTO-CASSETTE:
(THE AUTOMATIC CLIPPING SERVICE FOR TV NEWS)

by

Aya Konishi

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1986

Copyright (c) 1986 Massachusetts Institute of Technology

Signature of Author _____
Department of Electrical Engineering and Computer Science
June 2, 1986

Certified by _____
Walter Bender
Thesis Supervisor

Accepted by _____
Professor David Adler
Chairman, Undergraduate Thesis Committee

MASS. INST. TECH.
JUN 18 1986
LIBRARIES

Archives

AUTO-CASSETTE:
(THE AUTOMATIC CLIPPING SERVICE FOR TV NEWS)

by

Aya Konishi

Submitted to the Department of Electrical Engineering and Computer Science on June 2, 1986 in partial fulfillment of the requirements for the degree of Bachelor of Science.

Abstract

This project describes the design and implementation of an electronic clipping service for broadcast television. The system, Auto-Cassette automatically records on video tape pertinent programming from various broadcast sources, and provides a rich environment for viewing this tape. In essence, the computer watches television for the user, electing to store those items of potential interest, and then provides to the user ways to randomly access information on the tape. This includes a "table of contents" of the tape. The user then selects from the table of contents and subsequently watches a sequential presentation. To give the project scope, input to the system is limited to news-related programming.

This work was sponsered in part by IBM Entry Systems Division.

Thesis Supervisor: Walter Bender
Title: Principal Research Scientist, The Media Lab, MIT

Dedication

To my parents.

Acknowledgements

I thank members of the Media Lab, especially the NewsPeek group, for providing help and support for my project. I'd especially like to thank my thesis advisor, Walter Bender, who had given me so much time and effort.

I also thank my friends, Bill Saphir and Pat Jennings, for always being there when I needed advices on my programs.

This work was sponsered in part by IBM Entry Systems division.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Project Overview

The Auto-Cassette project involves the design and implementation of an electronic clipping service for broadcast television. The system automatically records on video tape pertinent video programming from broadcast sources, and provides a rich environment for viewing this tape. In essence, the computer watches television for the user, recording those items of potential interest on tape, and then provides to the user ways to randomly access this information at the "end of the day". This includes the creation of an on-line "table of the contents" for the tape. The user then selects from the table of contents and watches a sequential presentation. The project has been limited to news-related programming.

Auto-cassette provides a time efficient way of viewing the television news. As the result of the system, the edited program does not contain information that the user does not want to see, nor does the user have to wait for the information of interest. The personalization process is very important in an age when there is too much information available for any one person to absorb.

1.2 Design Overview

The hardware environment of Auto-Cassette is an extension of the typical household television system; local storage and computation have been added. The broadcast is no longer directed at the television receiver, but rather at the home computer. The local storage enables one to view the broadcast asynchronous to its reception. Computation is used in both the storage and the retrieval of video information. A video tape recorder is

used to store broadcast television. The computational component is used to selectively record incoming material and to facilitate subsequent viewing.

A close-caption decoder has been modified to provide its output to the computer rather than to a character generator. Hence the computer is provided with an ascii transcript of the broadcast. This data, along with the television schedule, is used to determine detailed information about the programming; e.g. we know not only that we are watching ABC Nightly News, but that the lead story is about President Reagan's reaction to the current Middle East crisis.

The ascii transcription is passed onto the Edit system. The Edit system functions as a filter. Sequences which are of interest to the user are identified by a boolean operations on keywords. Stories are sorted by an algorithm which takes into account order of transmission (the generally most important stories are broadcast first), and the result of the filter mechanism (stories which match closest to the user profile are given priority in the presentation).

No matter how well the Edit system performs, it will always lag somewhat behind in its perception of the interests of the user. Hence it is crucial to provide a table of contents and a method of resequencing the material. Two methods of presenting the contents of the tape were proposed initially. Only one was implemented within the context of this thesis. The first method is to create a table of contents by extracting headlines from the stories, along with a still frame illustration. The second method is to create short video sequences for each news story which give the flavor of the story, but not any of the details. These sequences would be reminiscent of "News Break," the short interruptions to the regular broadcast which the local stations use to inform their audience as to what will be featured in that evening's new broadcast.

The table of contents is used to refine the editorial process, i.e. to select a subset of what is available and to resequence these selections. The input mechanism is a "touch sensitive"

display. Touching a headline causes that story to be queued for subsequent viewing. After the selection process, the video is played sequentially.

Chapter 2

Design

2.1 Overview

The input to the Auto-Cassette system is the ABC Nightly News, broadcast daily at 7:00PM. This news program was selected because it is currently the only national network news broadcast with closed-captioning. The news is recorded on a video tape, and text from the closed-caption is stored on a magnetic disk. The closed-captioned text is a transcription of the audio on the tape. By making use of this data, the system then provides a interface to the news items on the video tape.

Functionally, the system can be divided into three sections.

1. Capturing the broadcast news onto video tape (Capture)
2. Making a database of stories (Edit)
3. Presenting information on each stories (Selection & Playback)

Each of three modules is maintained separately. Each section is explained in detail below.

2.2 Capture

The first section of the system deals with the retrieving and storing of the database. The environment is an extension of the typical household home video system. The hardware consists of television receiver, a computer-controlled video recorder, a closed-caption decoder and a personal computer.

Simultaneous to the broadcast, the system records the video portion of news onto a video tape and the captioning portion of it on the hard file of the personal computer. Timecode is stored along with the captioning in order to maintain a correspondence between the video and the transcript. The captioning is then filtered and the text of each story (news item) is

put into separate file. The system also creates an index file where the name of the text file and the beginning location and the ending location on the video of all the stories are kept. In the second pass over the tape, the system grabs a single frame of video from each story, and stores the image on the personal computer.

2.3 Edit

After the news has been captured and partitioned into individual stories, the editing process begins. This sub-system functions as the primary filter of stories and as a sorter. The Edit system only selects those articles which match the user's interests to pass onto the presentation system.

The Edit system makes use of information regarding what kind of articles are of interests to the user. This information is maintained in a keyword file, provided by the user. The words in the file are matched against the text of the news stories. Each line of the file contains a (set of) word(s), to be matched. If a story contains any keyword, it is passed to the Presentation system. In the absence of the keyword file, the user is presented with every story.

This method of filtering was chosen for two reasons. It is simple and efficient to implementation. Word-matching takes relatively short time to run. Also, it is easy to modify. To change the filter, one need only to add or eliminate keywords from the file. On the other hand, research in this area indicates that this method is rather inefficient.[1] Editing could be improved by application of more intelligent modeling of the stories, but this is outside the scope of this thesis. Other methods are discussed in Chapter 4.

Another function of the Edit system is sorting of stories. The information accumulated by the capture system must be put into a data structure which can be accessed efficiently by the

---

[1]See [Blair 85].

Presentation system. The Presentation system needs the following information for each story: a headline, the location of the representative still picture, the date of transmission, and the location and length of the story on the tape. All of the above information is computed or collected during the editing process. The main advantage to doing this compilation before hand rather than during the presentation, where the data is used, is time efficiency for the user. Since the presentation is visible to the user, any computing that can be done before hand should be done. Also, if the specification of file format changes, only the Edit system needs to be modified. Changes need not propagate to the presentation.

2.4 Presentation

The Presentation system is the only sub-system of Auto-Cassette with which the user interacts directly. The presentation generates a the Table of Contents from information provided by the Edit system. Ideally, the Edit section should provide a pre-edited tape, which is sequenced according to a good guess of the user's interests. In this project, this method was not realized because of the inability of our hardware to edit the tape. Several other system limitations influenced the design of the Presentation system. Both the rendering of the table of contents and searching for stories on the video tape are time consuming. One design criteria was to minimize the impact of delays.

Since the matching of user's interests and stories done in the editing section can not always correlate with the user's current interests, the system must also provide a method for a user to resequence the stories. Therefore, the main functions of the Presentation system are displaying the contents of news items, letting the user select what to watch, and playing the video of selected stories. There are many ways to go about doing these. One way is to Playback each story immediately after selection by the user. Hence the user selects a story, then the system plays that story back, then the user selects another story and the system plays that one back, and so on. Control goes back and forth between the selection function

and Playback function. Another way is to let the user select all the stories at the beginning, and they play them back in the order he has chosen. This way, the control of the program goes only once to the selection section and then to Playback section. It was decided to use the latter approach because of time efficiency in switching back and forth between the selection and Playback sections.

2.4.1 Selection

Two methods for a user to select and resequence stories were proposed: the Table of Contents, and the NewsBreak. With the Table of Contents approach, the information about each story is displayed to the user in a way reminiscent of the 'contents' page in a magazine. The NewsBreak approach would show an abbreviated video clip of each story. Only the Table of Contents approach was implemented because of anticipated problems with the NewsBreak approach.

The input to the selection section is the file created by the Edit system.

2.4.1.1 The Table of Contents Approach

The approach which was implemented is an interactive table of contents. On a TV display with a touch-sensitive device, the system displays a one or two sentence headline, a "related" still image, and an indication of length of each story. The headline, the still image and length are all passed on from the Edit system. There are usually four article on one screenful. Since the entire table of contents will not fit on one screenful at once, a paging mechanism is used. The screen is diagrammed in Figure 2-1.

At any time, the user by touching the portion of the screen, has the options of either selecting a story for viewing, going to the next or the previous page, or ending selection process.

When the user touches either the headline or the "related" still image of a story, that story is 'selected'. The color of the headline is changed to indicate the reception of the 'select'

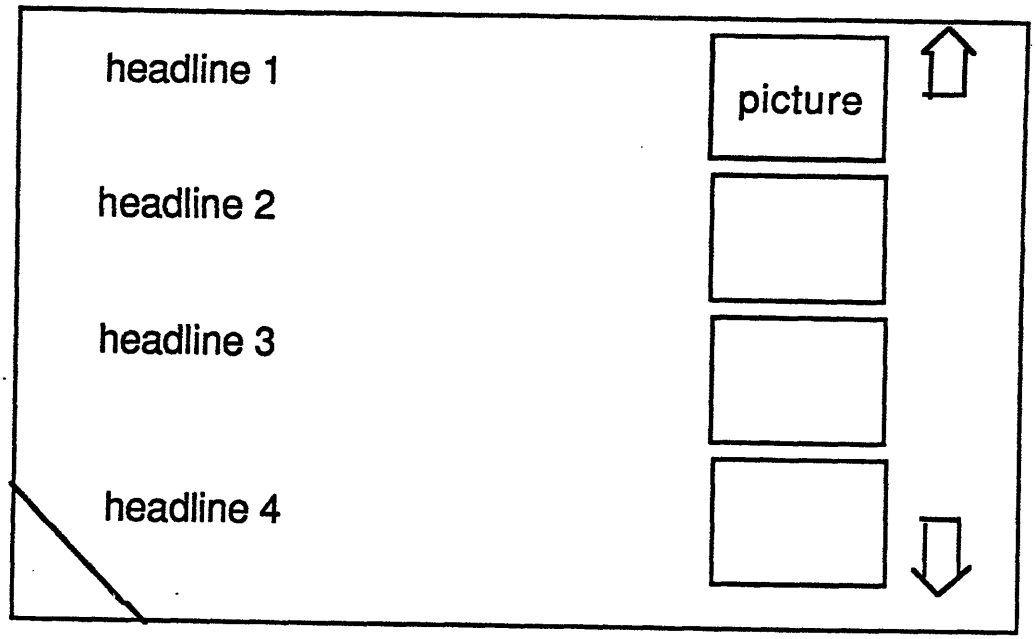


Figure 2-1: Selection section in progress

command. The user may touch the same story more than once, but the touches after the first one are ignored and would not be registered. The touches in the "invalid" region are also ignored. The system keeps getting the input until the user indicates the "end of input" by touching the appropriated section on the screen. Both the previous-page command on page 1 and the next-page command on the last page, are ignored. Figure 2-2 shows the valid area to touch for each command.

The order of presentation (of selected stories) is the same as the order in which the articles were selected. After the user indicates the 'end of input', the selection section ends and passes the flow to the presentation section. The Playback section plays the video of the selected stories.

2.4.1.2 The News Break Approach

The NewsBreak approach is to show a tightly edited video sequence of each story included on the tape. This is intended to be similar to "News Break", a preview of the evening's news used widely by local broadcasters. In this approach, the user is shown a sequence of short video segments which are a portion of each story which has been recorded. The user,

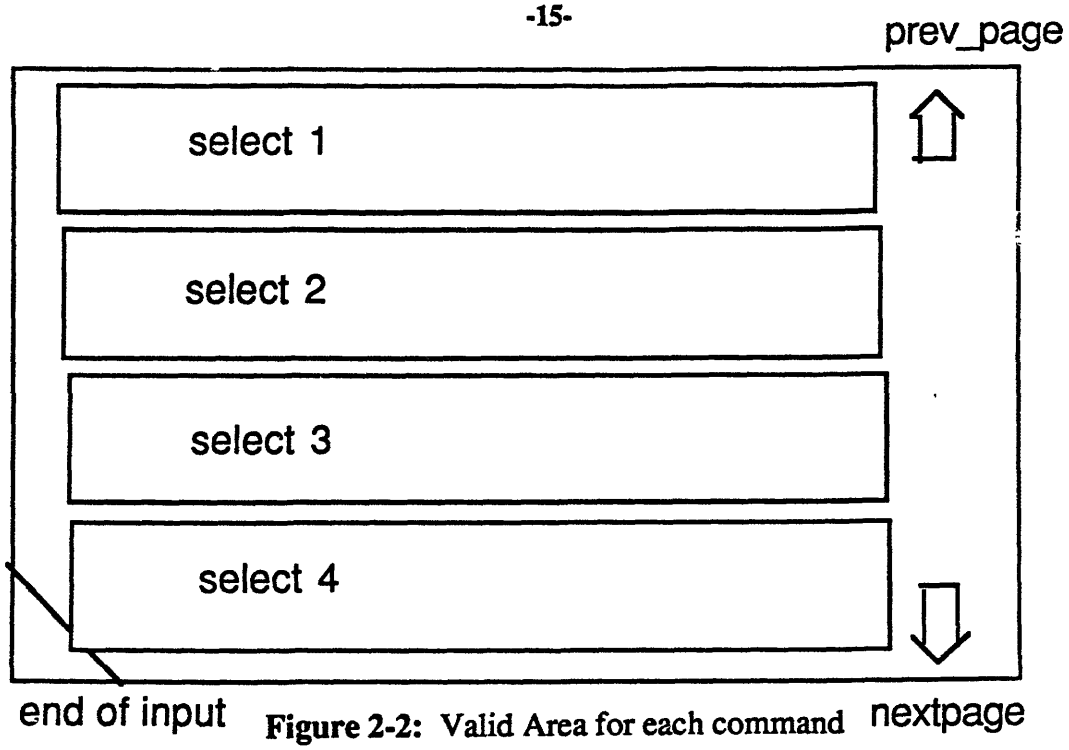


Figure 2-2: Valid Area for each command

as he watches each item in the sequence, can decide whether or not he would like to view the entire story.

This approach has an advantage over the Table of Contents in that it provides much more information about the stories. Even 10 seconds of video is likely to contain much more information than a headline and a representative still image. A video is also more visually attractive and entertaining. But it is very hard to re-edit already tightly edited video. For example, it would be difficult to stop the video without cutting off speaker's voice abruptly. It is unclear that the tightly edited broadcast lends itself well to re-editing. Also, since the video is sequential rather than parallel access, NewsBreak requires more time on the user's part. If we only show 30 seconds of each story, to watch NewsBreak of all the stories (~20) would take 10 minutes, which is too long. The Table of Contents is inherently parallel accessible. Because of these reasons, the Table of Contents approach was adapted and implemented.

2.4.2 Playback

Since the stories are not necessarily sequential on the tape, the tape player must search before playing each story. Stories are played back in selected order. Ideally, the system should have two play back decks, so that one deck can be searching for next story while the other is playing current story. Our implementation only used one tape deck, and therefore must account for the search delay. While the tape is searching, the illustration and headline of the next story, which were used in the table of contents, are displayed.

Chapter 3

Implementation

The system was implemented on a SUN Microsystem Network at the Media Laboratory. The software environment was Berkeley UNIX version 4.2 and the "C" programming language. The hardware environment included a Datacube frame buffer, an AMPEX video tape recorder, an Elographics touch sensitive device, and a Grass Valley video switcher.

3.1 The Top Level Implementation

The input to the system is the television broadcast and the output is a video tape with a table of contents and an interface to watching it. It is divided into four modules: capture, edit, selection and Playback. The Figure 3-1 is a the block diagram for the modules.

In the subsequent chapters, each module is explained in detail.

3.2 Capture

3.2.1 Hardware

A computer-controlled video recorder is used for the capture and play back of the television signal. A closed-caption decoder has been modified to divert the captioning to a serial port on the computer. Still images are grabbed from the video by a frame-buffer with an acquire cycle. The same framebuffer is subsequently NTSC encoded and used for the Table of Contents display. A video switcher is used to move between the encoded frame buffer output and the video tape recorder.

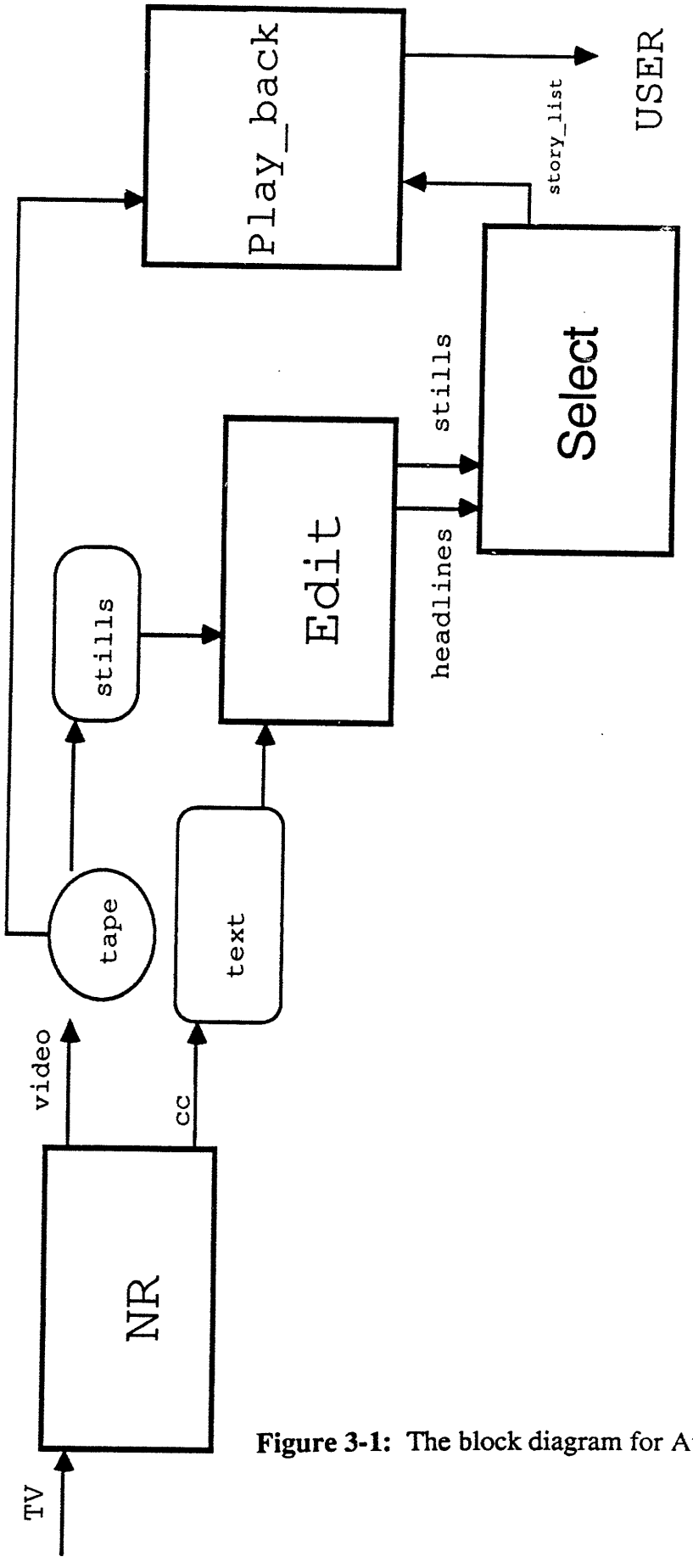


Figure 3-1: The block diagram for Auto-Cassette

3.2.2 News Retrieval System

The News Retrieval System was developed at the Architecture Machine Group at MIT.[2] Its input is a television broadcast and its output is a recorded video tape and text and image files stored on the personal computer. The text files contain the closed-captioning portion of the broadcast. The captioning has been partitioned into individual stories. The system detects the beginning of a new story by locating story delimiting characters inserted by the broadcaster. The video portion of the input is recorded on a tape. A heuristic is used to find relevant stills. Half way into each story, the system grabs a picture from the video and stores it as a bitmap on disk. The block diagram of the system is shown in Figure 3-2.

3.2.3 The Capture System

This section retrieves the information stored by the News Retrieval System and puts it in a data structure easily usable by the "Auto-Cassette" system. The section consists a single module which has one procedure, **create_article_list()**.

```
main(argc, argv)
{
/* Main procedure for capture&edit sub-system.
 * Calls create_article_list() and make_toc().
 */
}

struct article article_list[100];
create_article_list(date)
int *date;
{
/* This module takes the information from date.index
in ~vmb/news/stories, and creates a data structure
articles article_list[].
The article_list[] declared externally. */
}
```

create_article_list() takes as it's input data captured by the News Retrieval System, stored in the index file (*date*.index), and puts it into a data structure, *article_list[]* for use by the edit system. The specification of *article_list[]* is as follows.

---

[2]See [Bove 83].

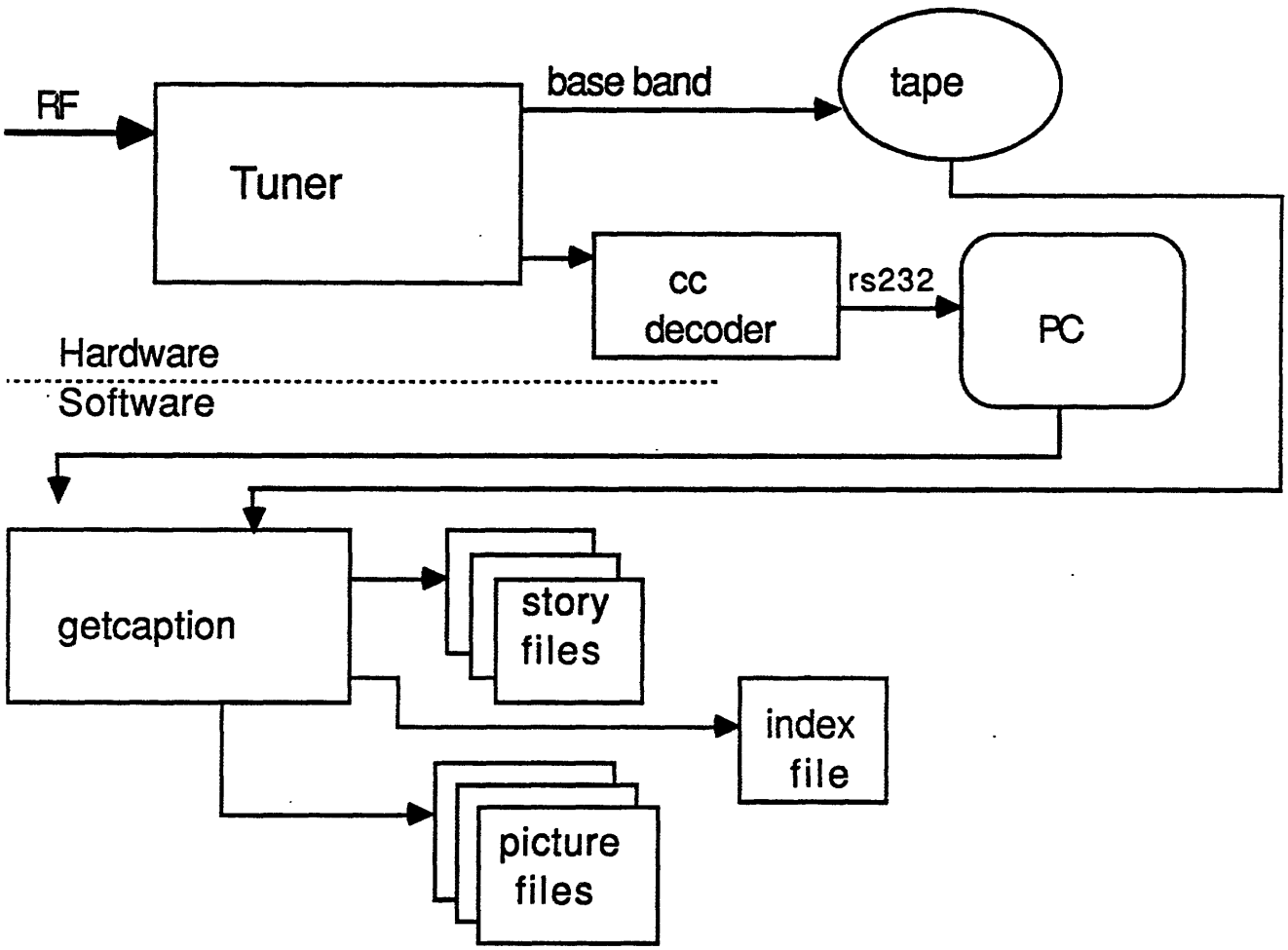


Figure 3-2: The block diagram for the NR system

```
struct article {
        char text[40];
        char image[40];
        char date[20];
        int  location;
        int  length;
}
```

The *text* and *image* fields of an *article* are UNIX file names. These files contain the text of the story and a bitmap of a "representative" still picture from the story. The *date* field is not really necessary at this time, since all of the articles are from the same tape. It is included for possible future extension of the database to multiple dates and sources. The *location* field represents the offset in frames of a story from the starting time code on the video tape. The *length* is the length of the story in frames.

3.3 Edit

The Edit system has two functions: matching interests, and headline extraction. The main procedure, **make_toc()** takes *article_list[]* as (external) input, and creates a *toc[]*. The data structure, *toc[]* is defined as follows.

```
struct toc_entry {
        char headline[1000];
        char image[30];
        int  location;
        int  length;
} toc[100];
```

The *toc[]* contains only stories which match the users' interests. The most of the entries are copied from the *article_list[]* except *headline*.

Figure 3-3 shows the module dependency diagram for the editing module. **make_toc()** which is the main procedure in this cluster, calls two sub-modules; **match_interests()** and **grab_headlines()**. **make_toc()** first runs **match_interests()** on each *article* in the *article_list[]*, and if it matches one of the user's interests, it calls **grab_headlines()** on the *article* and puts the result in *toc[]*. **Match_interests()** and **grab_headline()** both call sub-procedures to complete their tasks.

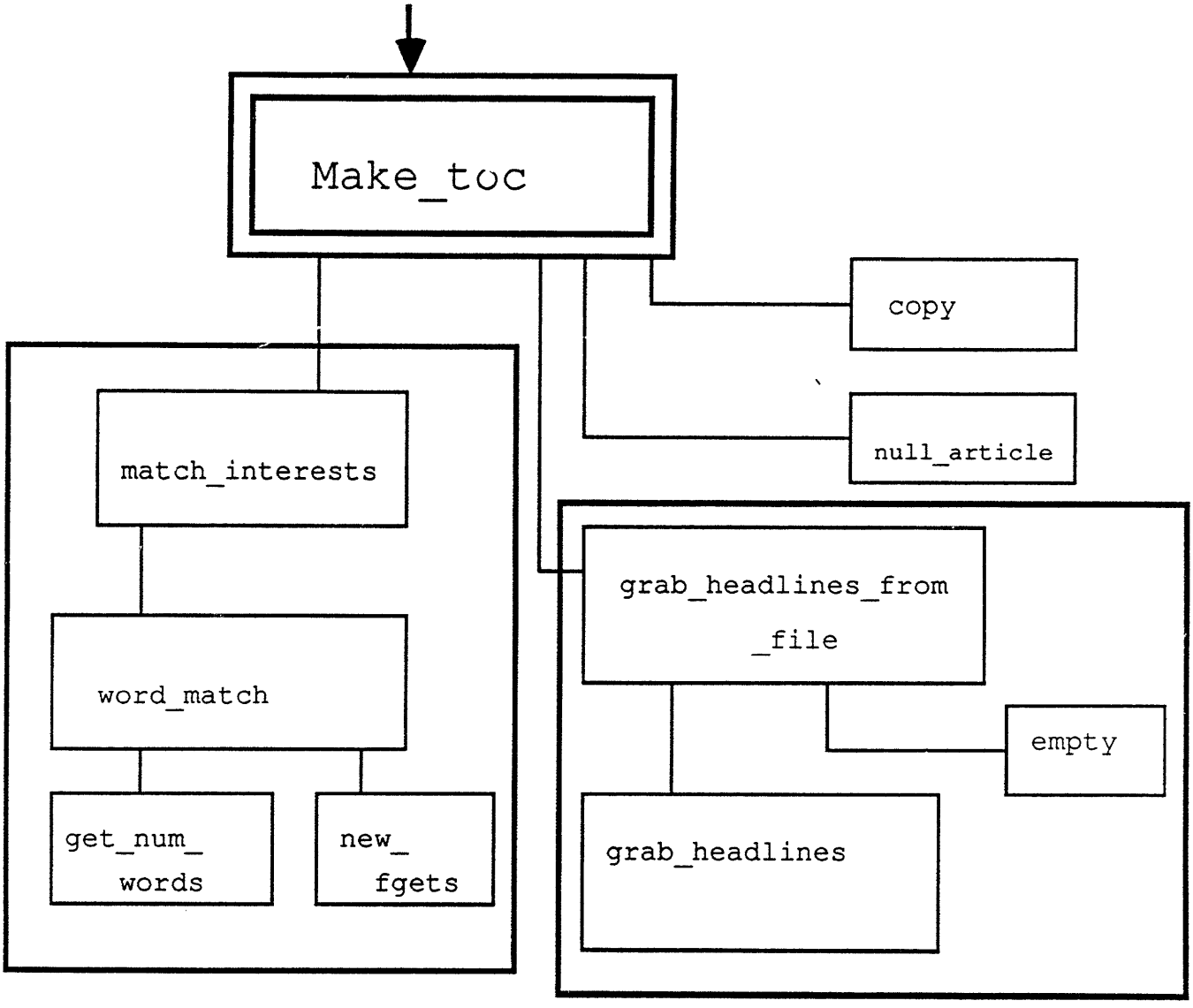


Figure 3-3: The module dependency diagram for the editing section

3.3.1 Matching Interests

Matching is done in a procedure called **match_interests(filename)**. The list of interests are provided in the file "interests.lst". This procedure

calls **word_match(text, file)** which does the actual work of matching words. **word_match(text, file)** is used in the greetings elimination process also.

```
match_interests(filename)
char *filename;
{
/* This procedure opens the file, and calls
   word_match(text of the file, "interests.lst".
   If word_match returns 1 or -1,
   match_interests returns 1, meaning "include this story".
   Otherwise, it returns 0 meaning "don't include this story.".
*/
}

word_match(text, file)
char *text, *file;
{
/* This procedure checks if any words given in the
file is contained in the case that there is no words
in the file, or the file is not found.
It is case sensitive, and ignores any preceding and dangling
blank spaces of any line.
  RESULT (integer) :
        1   there is a match
        0   there is no match
        -1  File not found
*/
}
```

The **word_match** works as follows. The **file** contains the sets of words which would be matched against the **text**. Each line of the file denotes one set of the words, and it can contain any number of words. Each set is treated as a group and all the words in a set must appear consecutively. For example,

```
THE GOVERNMENT OF WEST GERMANY TODAY EXPELLED 2 LIBYANS FROM THEIR
EMBASSY IN BONN IN WEST GERMANY.
```

This sentence would match with "GERMANY" and "WEST GERMANY", but not with "WEST GERMANY POLICY" or "EAST GERMANY".

3.3.2 Headline Extraction

The other editing function is to extract a headline from the text of the news stories. The news stories on the broadcast TV news are already edited tightly, and they closely follow a format of: (greetings), headline [by the anchor man], body of story [by a reporter]. The system takes advantage of the rigid formatting. The first non-greeting line is considered to be the headline. The greeting is recognized by using the **word_match(text, file)** procedure. file, in this case, is a file which contains possible greetings such as "Hello" and "This is the Nightly News". This scheme works well. Here is an example of a typical story and its headline.

```
> THIS IS JOHN MCWETHY.
  THERE ARE TWO AMERICAN
  AIRCRAFT CARRIER TASK FORCES
  IN THE MEDITERRANEAN.
  BOTH HAVE HAD THEIR ORDERS
  CHANGED IN THE LAST
  24 HOURS.
  THE "CORAL SEA" WILL NOT BE
  HEADING FOR HOME.
  THE "AMERICA" WILL NOT MAKE
  A PORT CALL IN FRANCE.

  .........

  JOHN MCWETHY, ABC NEWS,
  THE STATE DEPARTMENT.
```

The headline extracted was:

```
THERE ARE TWO AMERICAN
AIRCRAFT CARRIER TASK FORCES
IN THE MEDITERRANEAN.
```

3.4 Selection

The "Auto-cassette" system adapted the table of contents approach for its basic input method. Two of the reasons are its straight forward implementation and its ease of use. It's main function is **type_set()** which takes the *toc[]* (kept in an external variable) as input.

The **type_set()** module is divided into two sub-modules. One module, **display_page(page number:int)**, takes care of updating the screen, and the other, **get_touch_input()** takes care of getting touch inputs from the user. These modules interact frequently because each input causes the screen to be updated in some manner. They also have to be running concurrently to allow the user to input commands even while the screen is being updated.

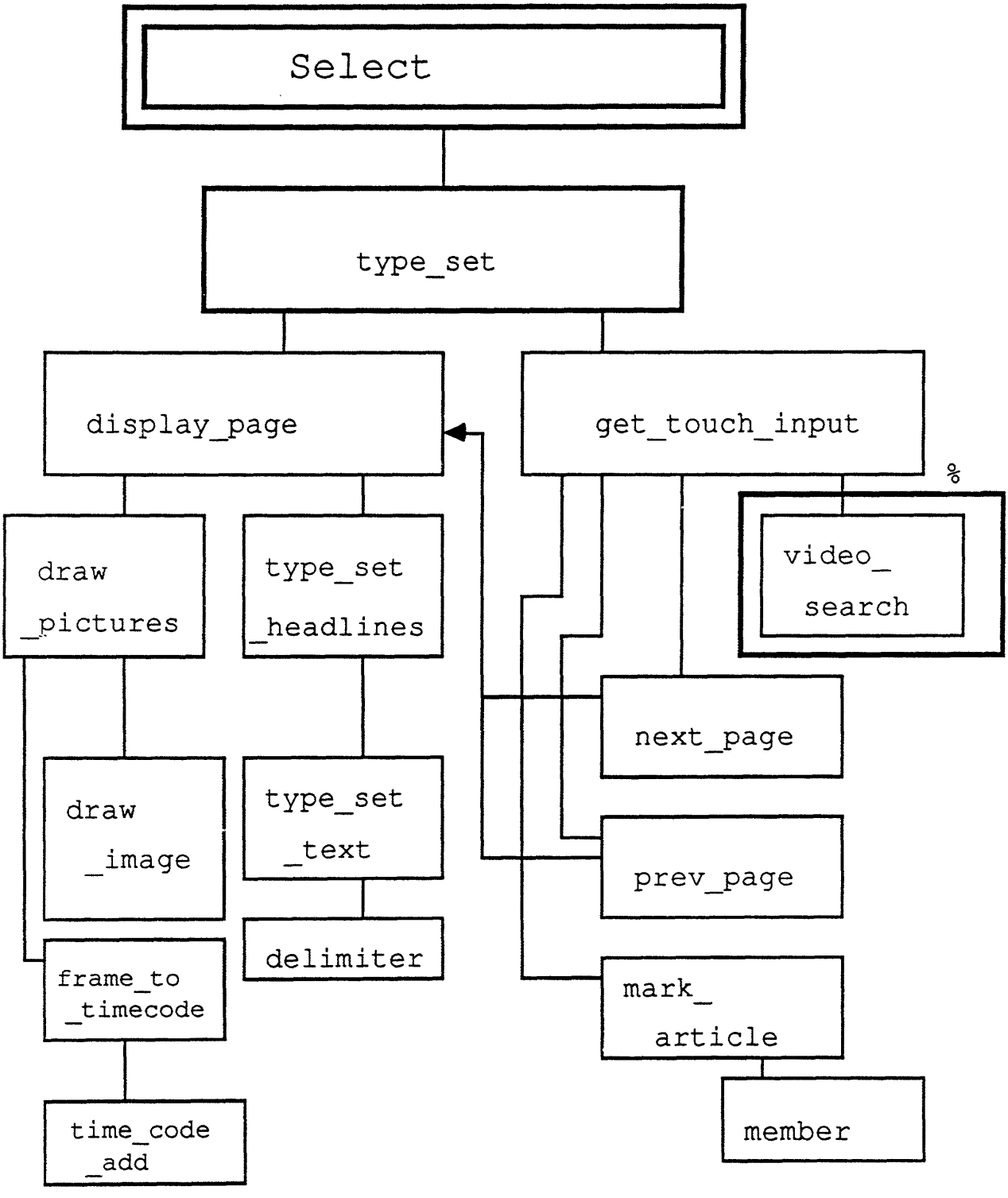
type_set() calls **display_page(*1*)** in order to draw the first page of the Table of Contents. Then **get_touch_input()** is called until the interaction with to the Table of Contents is completed.

Each of the sub-modules are explained in detail below. See Figure 3-4 for the module dependency diagram for this section.

3.4.1 Display_Page

The **display_page(page_number)** is responsible for drawing a single page of the Table of Contents. The page number is passes as an argument, while the number of stories and the number of story headlines per screen are set in the external variables, *num_articles* and *num_div*. Using this information and an array, *toc[]*, **display_page()** renders the screen as was described in Chapter 2.

The task of updating screen is divided into two parts: the image portion and the text portion. Accordingly, two sub-procedures were implemented: **display_pictures(page_number:int)** and **type_set_headlines(page_number:int]**. **display_page()** forks and creates two processes, each running one of the sub-procedures. The main reason in dividing the task is



% -- This procedure belongs to play_video_module

Figure 3-4: The module dependency diagram for Selection section

the time efficiency. Since the drawing of pictures is time-consuming (1 to 2 seconds for each still), if it is done sequentially, the user would have to wait before he can either 'read' the next headline or select a story. On the other hand, this implementation allows input at all times. Because **type_set_headlines()** returns immediately and passes the control of the program back to the **get_touch_input()**, the two procedures which serve the main functions of selection section, **type_set()** and **get_touch_input()** run concurrently.[3].

type_set_headlines() extracts the *headline* field of each *story* in *toc[]* and type_sets it on the screen. In this implementation, four *story*'s are displayed on the screen, but the number is arbitrary, and can be set in the include file "autocassette.h". This number is chosen because it seemed to be the best considering the size of the picture and the text. The colormap used is specified in 'news.cm'. The text is written on the screen using the soft-fonts (2 bits per point) on which four shades of color is used to display text in stead of one. The result is a better visual quality of video text displays.[4] In Auto-Cassette, five sets of four entries in the colormap are allocated to specify the color of each headline. Initially, the text is written in a shade of blue. The sub-procedure **type_set_text**(*text, x-orig, y_orig, width, length*) type sets *text* in a rectaliner region specified by *(x_orig, y_orig, width, length.* **type_set_headlines()** also displays the time of each story the left corner of the illustration. Time is calculated from the *length* field. This calculation assumes the broadcast started at 7:00PM.

draw_picture() loads the picture into the framebuffer. One of its arguments is the name of the file in which the picture is stored.

---

[3]See Section 3.4.2

[4]See [Schmandt 83]

3.4.2 Get_Touch_Input

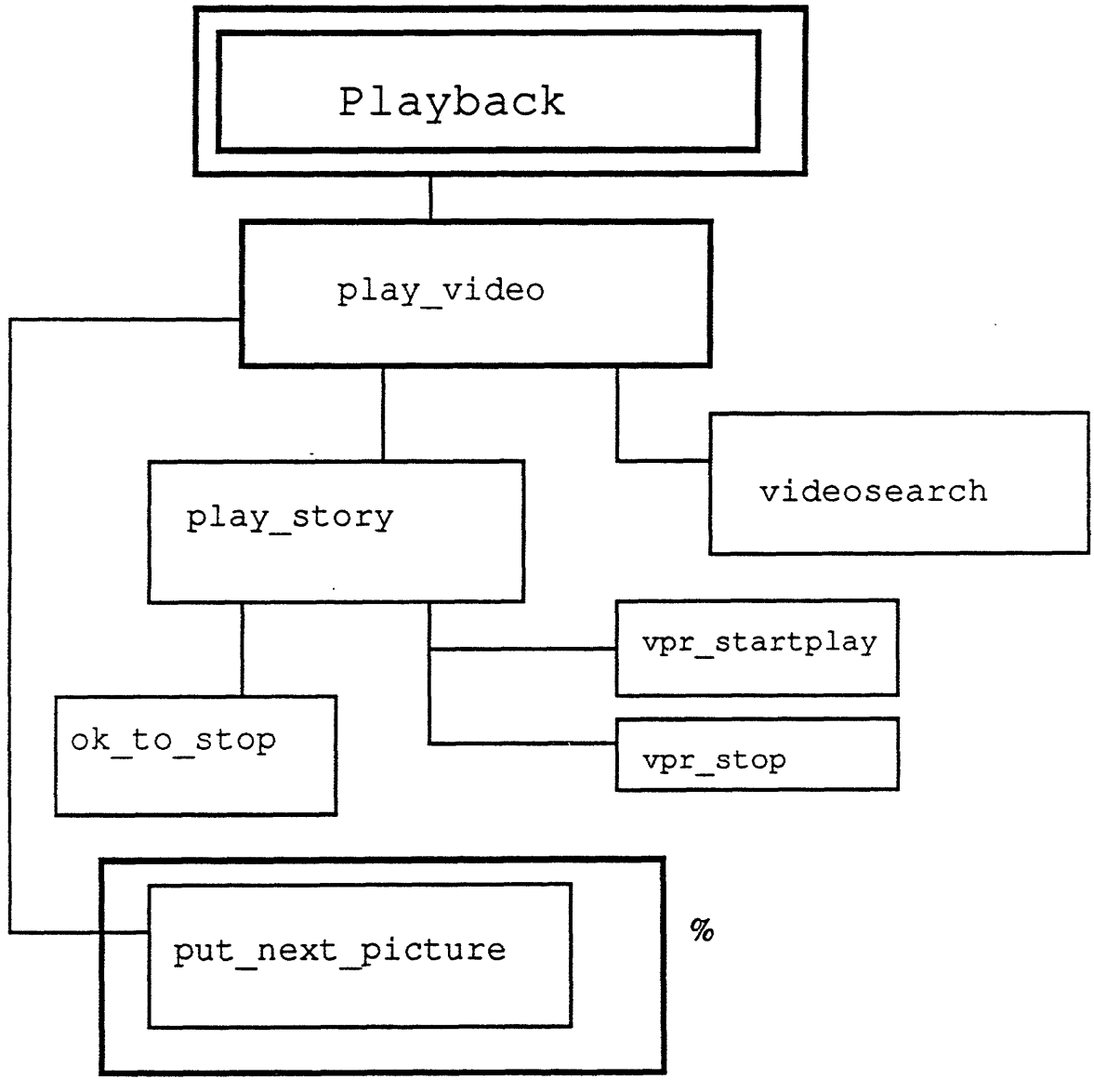
The **get_touch_input()** procedure is a loop which polls the touch screen, waiting for input from the user. On the screen, each region is assigned to a command. The procedure analyzes input from the touch screen and decides which command is being issued. It then executes the command. The loop is exited upon the receipt of the 'end selection process' command. The input can be one of the following: 1) go to next page, 2) go to previous page, 3) select this story, 4) end selection process, 5) invalid command. **get_touch_input()** calls **next_page()**, **prev_page()**, **mark_article()**. **exit()** or nothing respectively. The **next_page()** and **prev_page()** call **display_page(***page_number***)** with appropriate page number, which is (*cur_page* + 1) or (*cur_page* - 1). *cur_page* is an external variable which holds the value of the current page number. Upon receipt of the 'select' command, **get_touch_input()** first stores the number of the article selected in an external variable *int story_list[]*, if it is not already in the list, then calls **mark_article()**. **mark_article()** changes the entries in the colormap which correspond to the story being selected. The result is that the headline is changed from blue to red. Because of the separate addressability of the colormap and the bitmap, changing the color of text is independent from other activities on the screen. It can be concurrent with writing text or loading images. Hence, story selection can be instantaneous.

The interaction between the display and the input is complicated. Since **get_touch_input()** is running while the pictures are being drawn, it is possible to get input while the screen is not completely drawn. When either a 'next-page', 'prev-page', or 'end' commands is received, the system must terminate any on-going **draw_pictures()** processes before calling another instance of **display_page()**. To do so, the **get_touch_input()** remembers the process id of the on-going **draw_picture()** process. This information is passed to **display_page()** as an argument.

3.5 Presentation

At this point, the user, having selected the articles, watches the tape. The presentation module, **play_video()** is responsible for two things: controlling the video tape recorder and controlling the framebuffer. Since these two jobs must be dealt with concurrently, again, we use **fork()** system call to have two processes: one running **play_story()** and the other running **put_next_picture()**. . While **play_story()** is playing story N on video tape recorder, the **put_next_picture()** puts up a picture and headline from the next story (N + 1) into the frame buffer. When **play_story()** finishes playing story N, the display is switched to the frame buffer. By then, the picture for story N+1 has been drawn, so the user sees something on the screen right away. The frame buffer is displayed until the first process finishes searching for story N+1 on video tape. The display is then switched back to, playing the story N+1.

videosearch() is a sub-procedure called by **play_video()** to search for the next story on the tape. The search was done from the **play_video()** rather than in **play_story()** because there was a need to separate the searching for the next story, and the playing and stopping the video. Since there is no way to tell the video to "start playing and stop N seconds from now.", the system must constantly check to stop playing. That function is done in **ok_to_stop()**. **ok_to_stop** loops around infinitely comparing the current timecode and the timecode to stop until the difference is smaller than the specified accuracy period. The checking usually takes as long as 30 to 50 frames to pass by. This is the minimum amount of time allowed to lapse between each checking, considering the accuracy period of 5 frames, so one loop is executed right after another. While **play_story()** is waiting for **ok_to_stop()** to return OK, **put_next_picture()** draws the still image for the next story.



% -- this procedure belongs to type_set module

Figure 3-5: The module dependency diagram for Playback section

Chapter 4

Possible Extensions

4.1 Multiple Sources

The Auto-Cassette only deals with news from one showing of a news show. An extension would be to include news from multiple sources. This extension is easy as long as the multiple sources are not broadcast simultaneously. For example, the local news which precedes ABC Nightly News is also closed-captioned and could be used for a source for Auto-Cassette. These could be put on the same tape, and be treated as one large news show. It might be useful to cite the source of each news article. In that case, the programmer might want to add an extra field in both **article** and **toc_entry** data structures. A potential bottleneck is the amount of news that can fit on one tape. Capturing simultaneous broadcasts require a hardware extension. Two tuners and two decks would be required to capture both ABC and NBC news at 7:00PM.

4.2 History Mechanism

The Auto-Cassette system does not allow users to watch old articles. The system could be modified so that articles more than one day old are kept on the tape. The user would see the table of contents of the 'unseen' articles, but upon request, he can also see the 'already seen' articles. In this case, the date field in the **article** and **toc_entry** becomes very useful.

This leads to the idea of having a relational database. In this database, the articles of the same topic would be categorized together and sorted according to the time of transmission. In the table of contents, relations between stories would be indicated. The user might touch a region to specify that he wants to see the related articles, and smaller table of contents of

the related articles would be shown. Another approach is to let the user select a topic and to create separate table of contents of all the articles in the topic. This extension can be implemented but requires a substantial amount of work on the existing system. There must be a procedure which can figure out what categorize the articles. The maintenance of this database needs to be done at the Editing and Selection levels. The system would need to maintain the history of its usage, and must be constantly updated.

4.3 Further Enhancements

At this point, Auto-Cassette is very simple. The implementations as well as algorithms have been kept straightforward and simple. Below are some areas which would benefit from more sophistication.

4.3.1 Capture

- **Improvements on existing functions**
  The capturing system could be improved in several ways. One is in the delimiting of each news item. The beginning of each news item, determined by the NR system is not very accurate. It tends to be a little later than the actual beginning of the story. Another enhancement could be to distinguish news items from commercials.
- **Multiple sources**
  To be considered is the use of more than one source of information .
- **Other Databases**
  The input we can get from the TV broadcast is not only the text available. We can get sources such as wire services, as well as capture other video inputs.
- **Better picture selection**
  The selected illustrations are not necessarily representative of the news item. There might be a better way of selecting these pictures so that they tell us something more about the news item.

4.3.2 Edit

- **Interest Matching**
  The pre-matching of interests are done by a simple word-matching method. This method works well if the user's interests are very specific, but does not work well if the user knows the general area of his interests but not any specifics. Suppose he is interested in international affairs in general, but does not really want to specify any country names. Then, this method fails unless the text contains words such as "INTERNATIONAL" or "WORLD". An alternative method, is to have a set of key words for a topic. For example, any of "INTERNATIONAL", "SOVIET UNION", "LYBIA", "WORLD", "UNITED NATIONS" can be matched for the topic, international affairs. This will require substantial changes in **word_match()**.

- **Headline Extractor**
  The headline extraction is done by getting the first non-greetings sentence of the text from the closed-caption. It would be useful to construct a headline better than the ones used by the Auto-Cassette. This will require substantial amount of work in **grab_headlines()**.

- **More personalization**
  One way to give more personalization is to keep a record of what articles have been watched previously. Then the user would not need to see the an article on the table-of-contents after he watched it.

- **Edit deck**
  We can also create another video tape with only the articles which are pre-matched to the user's interests. This will eliminate the need for searching in the "play-back" section. But this change will require better understanding of the user's interests.

4.3.3 Selection

NewsBreak

The NewsBreak did not get incorporated in the Auto-cassette at this point, but it is an interesting idea to explore. See the design and the implementation section for details.

4.3.4 Presentation

- **Still image during searching**
  Auto-Cassette presents a still image and a headline of the next news item while it is searching for the next news item. Even though this is better than presenting a blank screen, it does not give to the user any new information. With a minor change in the **play_video()**, this space can be made useful. For example, it can show the length of the next news item and the expected length

of the search. Another idea is to indicate which section of this 'selected' version of news the user is watching now.

- **Adding Video Controls**
  Another feature that could be added is to provide video functions such as 'stop', 'restart', 'ff' while playing the video. We can split the output device's screen into two sections, one for playing video and and another for getting touch input for the video functions. To implement this, the **play_video()** needs to fork, and one process needs to do the basic video functions that **play_video()** does, and another process needs to get touch input. These two processes must interact with each other using interrupts. The system must keep external variables which contain information on where to stop, so that after being 'stopped' and 'restarted' by a user, it will still stop at the end of a news item and go to the next one.

Chapter 5

Evaluation

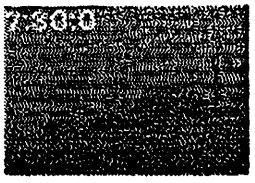
The Capture system seems to work well except that delimiting of stories is not precise. The system often would start playing a couple seconds after the story actually started, which can be annoying to the user. Otherwise, the capturing of closed-captioned text and still image work well.

The Edit systems' headline extraction is sufficient. Among 100 or so valid stories (not commercials) looked at, only few of them had a headline irrelevant to the story. The interests matching works well as long as the keywords provided are specific.

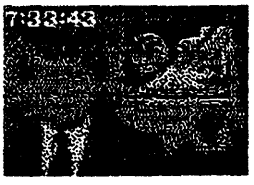
The Layout section's interaction with the user was most carefully designed and therefore proved to the best part of the system. Any touches were executed immediately. Figure 5-0 is an example of table of contents. It includes two screenfuls of table of contents for *The Nightly News* on ABC broadcasted on March 12, 1986.

The Playback section would have better interaction if the communication to video tape recorder were faster and more reliable.

16 MILES OFF THE FLORIDA COAST.



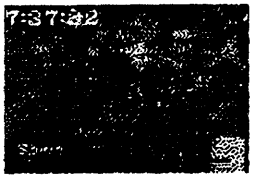
THE SUGGESTION THAT THE SHUTTLE "CHALLENGER" MAY HAVE BEEN LAUNCHED PREMATURELY BECAUSE OF GOVERNMENT PRESSURE GOT A NEW TWIST TODAY.



WHEN PRESIDENT REAGAN DELIVERED HIS STATE OF THE UNION ADDRESS, DELAYED A WEEK BY THE "CHALLENGER" ACCIDENT, HE EULOGIZED THE CREW OF SEVEN, INCLUDING SCHOOLTEACHER CHRISTA MCAULIFFE, THIS WAY -- WE PAUSE --

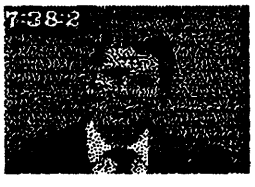
LATER IN THIS BROADCAST, AMERICAN CITIZENS HELPING THE SANDINISTAS.

RESENT PUBLIC OPINION POLLS HAVE SUGGESTED THAT WHEN THE VOICE OF THE PEOPLE WAS HEARD IN SPAIN'S ELECTIONS TODAY, SPAIN WOULD DECIDE TO DROP OUT OF NATO.



PRESIDENT REAGAN HE SENT HIS SPECIAL ENVOY PHILIP HABIB OFF TO CENTRAL AMERICA TODAY.

TODAY'S PHOTO OPPORTUNITY IN THE CONTRA AID BATTLE FEATURED PRESIDENT BIDING GODSPEED TO AMBASSADOR HABIB AS HE DEPARTED FOR CENTRAL AMERICAN TO PURSUE A DIPLOMATIC SOLUTION, SOMETHING MR. REAGAN'S CAPITOL --



WHO IS WINNING THE BATTLE FOR PUBLIC OPINION? OUR OWN ABC NEWS POLL TAKEN LAST WEEK FOUND THAT 59% OF THOSE RESPONDING OPPOSE NEW MILITARY AID FOR THE CONTRAS.

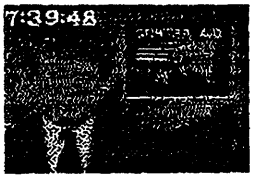


Figure 5-1: An Example of Table of Contents

Chapter 6

Appendix A

Include files

```
/*
 *   autocassette.h
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <local/tsd.h>
#include <local/dq.h>
#include <local/vpr.h>
#define STORYDIR "/u/vmb/news/stories/"
#define IMAGEDIR "/u/pix/"
#define SUFFIX "C.index"
#define IMAGEFILESUF ""
#define MAXLINE 1000
#define GBG 40
#define PLAY 10
#define STOP 11
#define REWIND 12
#define FF 13
#define NOCMD 14
#define ACCURACY 5
#define OK 0
#define NOTOK 1
#define ERR -1
#define NEWSBREAK 1

struct article { char text[40];
            char image[40];
            char date[20];
            int location;
            int length;
           };

struct toc_entry { char headline[1000];
              char image[40];
              int location;
              char date[20];
              int length;
```

```
            };

/*
 *      COLOR.H - used in display
 */
char *default_font = "cla17";
char *bigger_font = "cla20";
char *biggest_font = "cla20";
char *colormap = "news2";
int default_color[4] = {5,30,60,120};   /* WHITE */
int red[4] = {0,1,2,3};          /* RED */
int green[4] = {16,17,18,19};    /* GREEN */
int blue[4] = {48,49,50,51};     /* BLUE */
int purple[4] = { 80,81,82,83};     /* PURPLE */
int pink[4] = {112,113,114,115};
int yellow[4] = {96,97,98,99};
int paleblue[4] = {64,65,66,67};
int orage[4] = {32,33,34,35};
int text_color[5][4] = {{8,9,10,11},
                        {12,13,14,15},
                        {20,21,22,23},
                        {24,25,26,27},
                        {36,37,38,39}
                        };
byte cmvalue_b[4][3] = {{128,128,128},
                       {112,114,167},
                       {96,96,197},
                       {76,76,223}
                        };
byte cmvalue_r[4][3] = {{128,128,128},
                        {164,114,114},
                        {192,96,96},
                        {220,76,76}
                        };
```

Input Files

```
/*
 *  Greetings.lst
 */
GOOD EVENING
HELLO
GOOD AFTERNOON
GOOD MORNING
EVENING NEWS
```

References

[Blair 85] David C. Blair and M.E. Maron.
An Evaluation of Retrieval Effectiveness for a Full-Text Document Retrieval System.
Communications of the ACM , March, 1985.

[Bove 83] Victor Michael Bove.
A Flexible Integrated Framestore System.
Architecture Machine Group, MIT , 1983.

[Schmandt 83] Schmandt, Christopher.
Greyscale Fonts Designed From Video Signal Analysis.
Architecture Machine Group, MIT , 1983.