TV2Web: Generating and Browsing Web Contents From Video With Metadata

Mahendren MUNISAMY†, Kazutoshi SUMIYA†, and Katsumi TANAKA†

† Graduate School of Informatics, Kyoto University Yoshida Honmachi, Sakyo, Kyoto 606-8501,Japan

Abstract In this paper we look at automatically constructing web contents from video with meatadata. We refer to captured voice captions for each video shot as metadata. In this paper we reconstruct the metadata in accordance to the video image's data structure and propose a method to view (browse) the metadata together with the corresponding image data in a browser. Furthermore, using the "Zoom-In" and "Zoom-Out"metaphor, we introduce a method to seamlessly alternate between viewing video data and web contents.

Key words Video database, metadata, web contents

1. Introduction

Rapid progess in broadband and digital television technologies have made it possible to provide a vast volume of information to both internet users and television audiences.

As part of the progress in digital television, it has now become common to view broadcasts with voice captured captions (video metadata). In this paper we propose a method to view (browse) the metadata together with the corresponding image data after converting it into web contents. In [3] we converted video metadata into web contents. In this paper by implementing the "Zoom-In" and "Zoom-Out"metaphor, we introduce a method to seamlessly alternate between viewing video data and web contents. We also use the zoom metaphors to seamlessly alter the level of details (LOD) of the contents being viewed. The basic concepts of TV2Web are shown in Figure(1).

In order to acheive this, we first apply layered structures to the video data and extract the image data for each corresponding segmentation unit. Next, we reconstruct the metadata in accordance to the video images data structure. Finally, we use Java Script to automatically genarate web contents for the corresponding layers. The zoom functions are implemented by using Dynamic HTML (DHTML).

The follwoing section describes the approach we used to implement TV2Web. Section 3. explains how we generated the web contents to be displayed. Section 4. describes the functions available in our prototype system to browse the web contents. Section 5. describes the implementation of our prototype system, known as TV2Web. Section 6. concludes this paper.

2. Our Approach

We break down the video data as shown in Figure 2. Initially, the video data is divided into several video segments. Each segment represents a single semantic video unit. For example, within a newscast, each news clip would represent a single video segment. The video segments are then divided into video scenes. Finally each video scene is broken down into segments known as shots. A shot is defined as a continuous sequence of video images which have been captured by a single camera. A scene is defined as a continuous sequence of shots that captures a particular object (or group of objects) within the video data. The process of segmenting the video data was done with the help of methods introduced in [5]

We assume the following textual data is provided prior to implementing TV2Web:

(1) Title for video image

(2) Keyword: Each segmentation unit (segment, scene

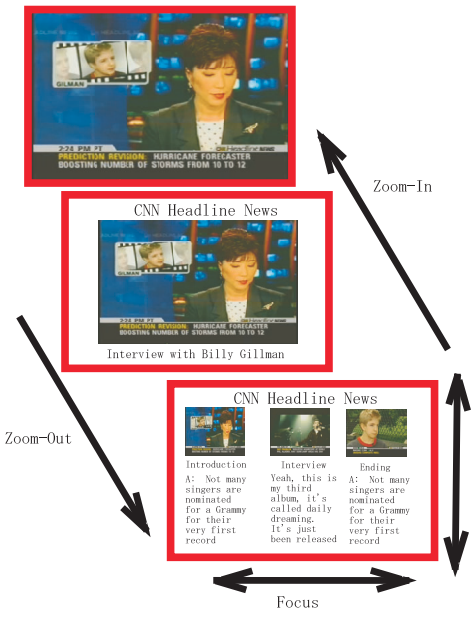


Figure 1 Concept of TV2Web

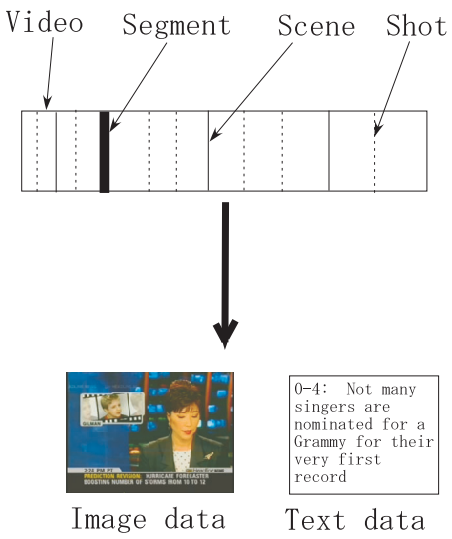


Figure 2 Break down of video data and data extraction

and shot) is assigned a keyword.

(3) Metadata: The metadata is restructured and mapped to the segmentation units (segment, scene and shot) it represents.

We also assume that each segmentation unit (segment, scene and shot) is provided with the following forms of image data

(1) Shot: 30 frames/sec video image.

(2) Scene: 15 frames/sec dynamic slide show.

(3) Segment: 5 frames/sec dynamic slide show.

(4) Video: Static key frame display.

In Section 3. we explain how we extracted and converted the data above into web contents.

3. Generating Web Contents From Video With Metadata

3.1 Obtaining Data

In TV2Web we presume that we are provided with segments of a video image. We used image recongnition software provided by [6] in order to obtain the images for the segments introduced in Section 2.. We assume that we obtain the images along with the absolute times those images were extracted from the video data.

When we developed our prototype system, we used voice captions provided during the broadcast of NHK's News7 program. These captions show the spoken content at the shot level and the absolute time each caption was obtained. It is therefore possible to obtain spoken captions for higher levels of the video data by simply merging the captions at the shot level. By using voice recognition techniques, it is also possible to obtain the voice captions for each of the segmentation regions mentioned in Section 2.. The voice captions for each segmentation region is stored as web text in a database.

The image data at the shot layer is represented by a 30 frames/sec video image. We use the original video data to represent the image data at this layer. The slide show images for the scene and segment layers are obtained by performing time based frame extractions on the video data. These frames are then displayed sequencially at the rates mentioned in Section 2. (15 frames/sec at the scene layer and 5 frames/sec at the segment layer).

Since we know the absolute duration of each shot's video image and metadata , the process of mapping the video image and metadata is done by simply matching the images and metadata that have the same time values. At the scene level, it is assumed that we obtained the absolute duration of the scene when performing the scene change detection on the original video data. As shown in Figure 2, since each scene is made up of several shots, we can obtain the metadata for a particular scene by simply merging the metadata of the shot's that make up that scene. This process is repeated for the segment and video layer as well.

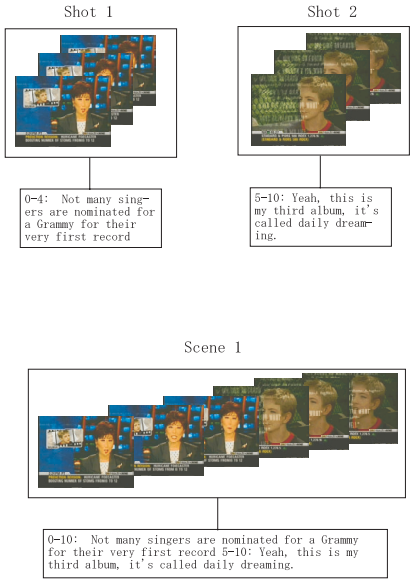


Figure 3 Generating web contents for TV2Web

An example of generating web contents for TV2Web is illustrated in Figure 3. We first the determine the duration of Shot 1 and Shot 2. Next, we break down the video image into the respective shots and obtain the spoken content of each shot. Next we determine the length of Scene 1 (which is made up of Shot 1 and Shot 2) and extract the appropriate frames that represent this scene. The metadata for Scene 1 is obtained by merging the metadata from the start of Scene 1 to the end of Scene 2.

3.2 Data Organization

We organize the image and textual data for each layer (shot, scene, segment, video) into blocks as shown in Fig-

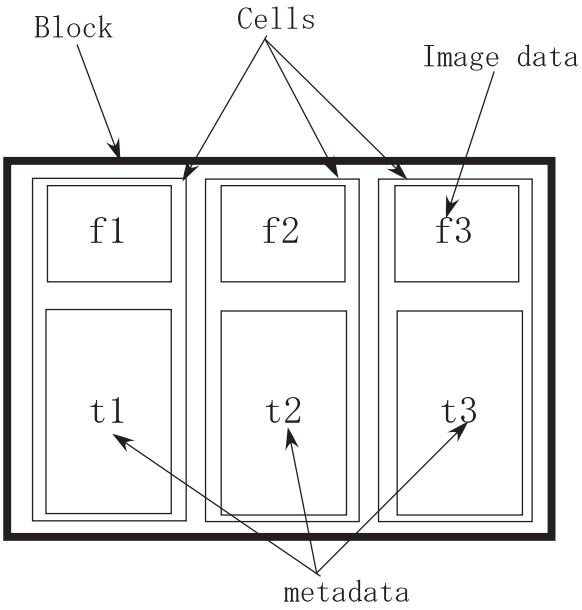


Figure 4 Implentation of video data structure

ure(4). Each segmentation region within a layer is represented by cells in Figure(4). Within each cell we place the respective image data ($f1, f2, f3\cdots$) and web text data ($t1, t2, t3\cdots$).

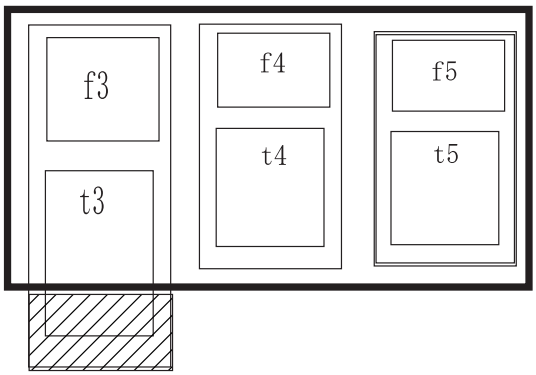
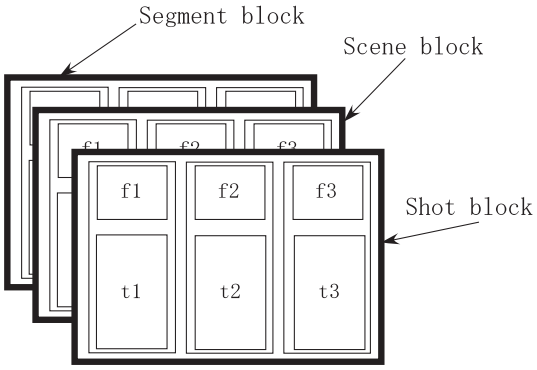


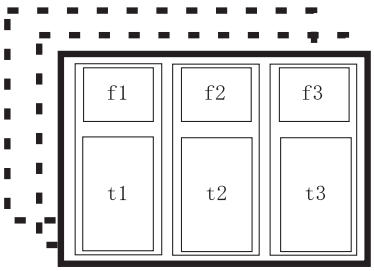
Figure 5 Cell too large for block

Only elements within a block will be visible during viewing. In other words, if the dimensions of a paritucar cell exceeds the dimensions of a block (shown by the stripped area in Figure (5)), the excess region will not be visible on the display.

All blocks that represent the various layers of the video data structure are given the same absolute positions (coordinates) when being displayed. As shown in Figure(6a). This means that all the blocks overlap on top of each other and this in return would make it difficult to view the respective layers. However as shown in Figure(6b), by manipulating the *visibility* property of each block, it is possible to display one block at a time. In other words, only the block to be viewed is made visible while the other blocks are hidden from view. This will allow us to implement the browsing mechanisms in



(a)Overlapping blocks



(b)Controlling block visibility

Figure 6 Resolving overlapping blocks

Section 4.

4. Browsing Mechanism

4.1 Zoom Functions

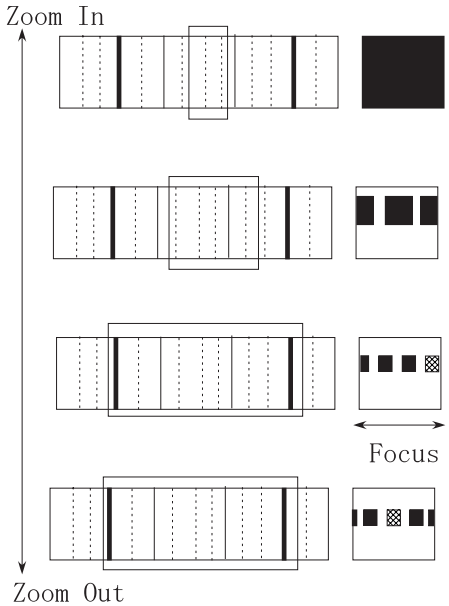


Figure 7 Zoom and Focus Funtions in TV2Web

Visualization of the various data structures introduced in Section 3. is shown in Figure(7). The Zoom-In function is used to convert from viewing web contents to viewing video data. This function gradually increases the size of the image data and gradually decreases the LOD of the textual data

being displayed. The zoom-out function is used to gradually decrease the size of the image data being displayed. This function also gradually increases the metadata's level of details (LOD).

4.1.1 Zoom-Out Function

Figure(8) shows the concept behind the Zoom-Out function. First the dimensions of the image data being viewed (box with no textures in Figure(8a)) are given an initial value. The algorithm for the zoom-out function is explained in the following steps:

(1) The visibility for all the blocks are set to *hidden*.

(2) The visibility of the block being presently viewed is set to *visible*

(3) The width and heigth (the dimensions) of all the cells and image data being presently viewed are decreased by a constant amount. In Figure(8a), the box wihout any texture represents the dimensions of a particular image data being presently viewed. As the Zoom-Out function is implemented, the dimensions of all the image data within the cell of a particular block are gradually decreased.

(4) The font size of the corresponding textual data is increased accordingly by a constant ammount.

(5) If the dimensions of the image data decrease beyond a certain threshold (shown by the box with the meshed texture in Figure(8a)) a new block is chosen to be viewed (Figure(8b)). The image data within the new block will have a smaller visible region and will be bounded by smaller dimensions. The textual data within the new block will have a higher LOD than the previous one.

(6) If the Zoom-Out function is still being implemented Return to step 1.

The algorithm for the Zoom-In function is similar to the Zoom-Out function except that in Step(3), the dimensions of the image data increase instead of decreasing. In Step(4), the font size of the metadata decreases accordingly during Zoom-In. Finally, in Step(5), if the dimensions of the image data increase beyond a certain threshold the new bounding dimensions will be bigger in the new block.

4.1.2 Zoom-In Function

Figure(9) shows the concept behind the Zoom-In function. First the dimensions of the image data being viewed (box with no textures in Figure(9a)) are given an initial value. The algorithm for the zoom-in function is explained in the following steps:

(1) The visibility for all the blocks are set to *hidden*.

(2) The visibility of the block being presently viewed is set to *visible*

(3) The width and heigth (the dimensions) of all the cells and image data being presently viewed are decreased by a constant amount. In Figure(9a), the box wihout any texture represents the dimensions of a particular image data being presently viewed. As the Zoom-In function is implemented, the dimensions of all the image data within the cell of a particular block are gradually increased.

(4) The font size of the corresponding textual data is decreased accordingly by a constant ammount.

(5) If the dimensions of the image data increase beyond a certain threshold (shown by the box with the striped texture in Figure(9a)) a new block is chosen to be viewed (Figure(9b)). The image data within the new block will have a bigger visible region and will be bounded by bigger dimensions. The textual data within the new block will have a lower LOD than the previous one.

(6) If the Zoom-In function is still being implemented Return to step 1.

4.2 Focus Function

While viewing the web contents, especially when the textual LOD is increased, it might be impossible to view all the contents on the same display. On a tipical browser, this problem is solved by implementing scrollbars. In TV2Web, we introduce the focus metaphor in order to overcome this problem. The focus function is implemented by pointing at a particular portion of the contents. This causes that portion to gradually move to the upper left portion of the display being viewed.

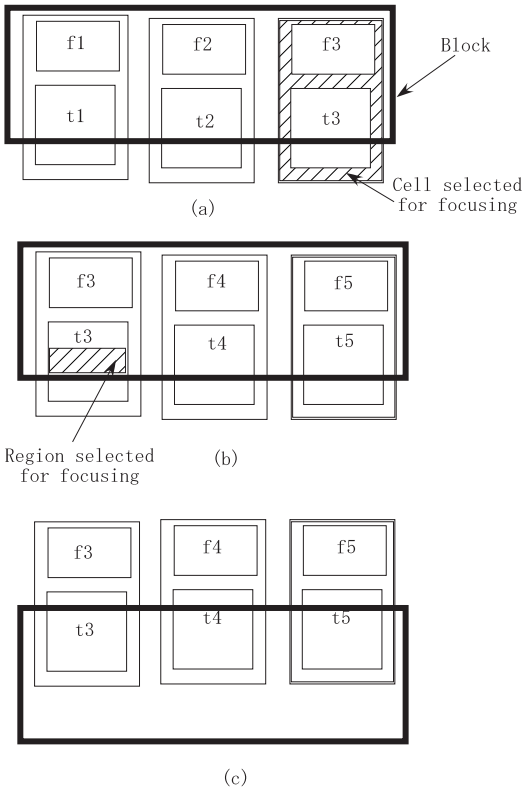


Figure 10 Concept of Focus Functions

Initially, in Figure(10a), the Focus function is carried out on an entire cell. This causes that cell to move to the left

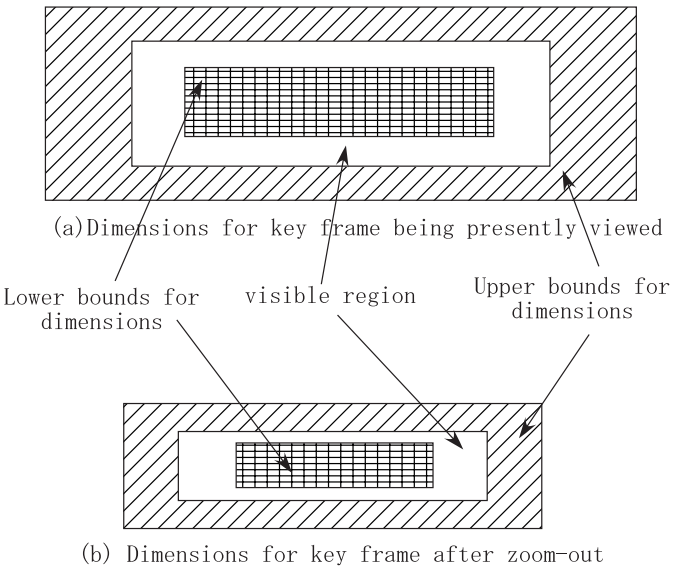


Figure 8 Concept of Zoom-Out function

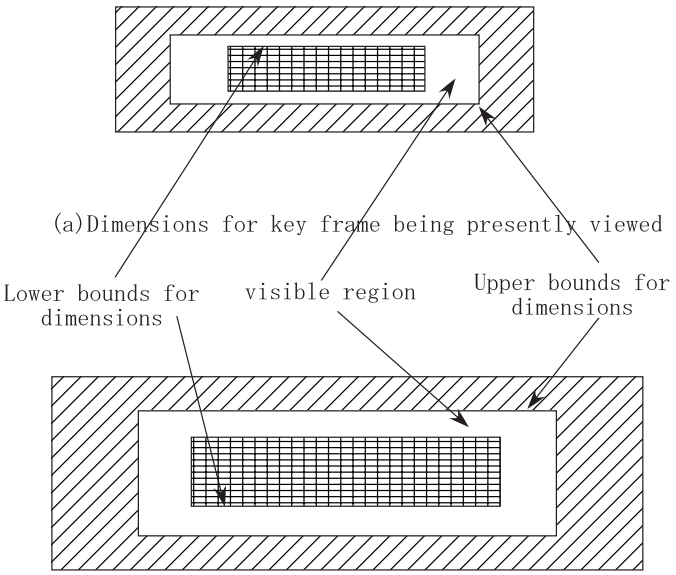


Figure 9 Concept of Zoom-In function

most portion of the block. The corresponding cells move accordingly. In Figure(10b) a subregion within a cell is chosen. The Focus function will move all that region and all corresponding cells to the the top left most portion of the block.

5. Implementation

5.1 Obtaining Images And Textual Data For TV2Web

We obtained the images for TV2Web by using Ricoh's MovieTool software. As shown in Figure 11, we are able to break down the original video data into the various segments introduced in Section 2.. The image data at the shot layer is represented by a 30 frames/sec video image. We use the original video data to represent the image data at this layer. The slide show images for the scene and segment layers are obtained by performing time based frame extractions on the video data. Using MovieTool, it is possible to perform time code based extractions both manually or automatically. We can therefore obtain the extracted images along with the relative time that they appear in the original video data.

As mentioned in Section 3., we organize the image and textual data for each of the structure layers (shot, scene, segmentand video) into blocks. In order to implement this, we first form an object called *cell* using JavaScript. The *cell* object consists of an array of images, $Images[1 \cdots n]$, a string constant, *metadata*, containing the textual data for each particular cell and a time constant, *time*, showing the

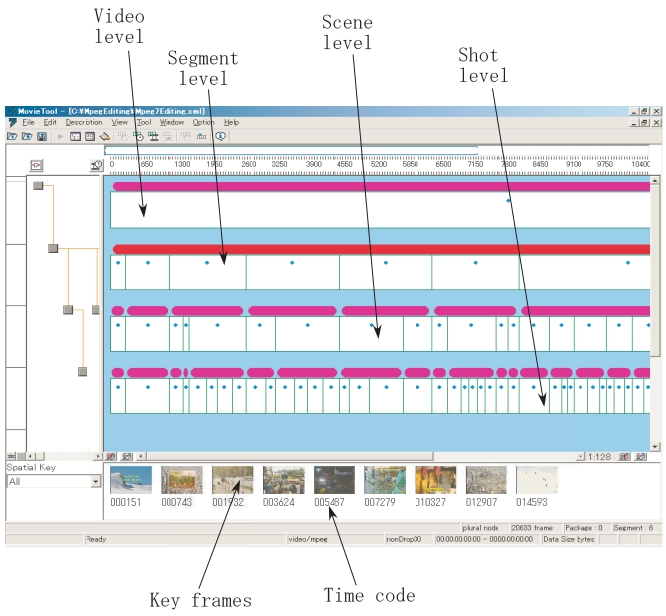


Figure 11 MovieTool

```
[001] 19:01:44 00104 一時帰国に向けて準備が始まっています。
[002] 19:01:48 00108 政府は、北朝鮮、朝鮮民主主義人民共和国
[003] 19:01:52 00112 による拉致事件で生存が確認された被害者５人が、
[004] 19:01:56 00116 今月１５日に、一時帰国することになったことを受けて、
[005] 19:02:00 00120 きょう関係省庁による対応チームを発足させ
[006] 19:02:04 00124 ました。北朝鮮による拉致事件の被害者の
```

Figure 12 Metadata sample

duration of each particular cell.

Figure 12 shows a sample of the metadata we obtained from NHK's News7 program. The metadata is comprised of four columns. The fist column is a sequential index of the metadata. The second column shows the relative time of the program while the third column represents the absolute time of the program. The final column is the captured voice captions that occured at the respective times.

As mentioned in Section 3., we store the textual data for TV2Web in a class called *textData*. We first read in the metadata in Figure 12 one line at a time as strings. We break each string down and the absolute time which is stored in the 3rd column is stored as an instance variable, *time*. The fourth column wchich represents the spoken content is stored as another instance variable called *caption*. This way we are able to automatically generate instances of each shot's textual data.

5.2 Implementation of Video Segments for Web Contents

The video data segments shown in Figure(2) is implemented in HTML using the $< DIV >$ tags. An explanation about $< DIV >$ tags is available in [2] A sample source code of this implementation is shown below.

```
<DIV ID="block0" STYLE="position:absolute;
left:0px; top:0px; width:720px;height:480px;
background-color:"#FFFFFF";
visibility="visible";> .....</DIV>
```

We implement the four data structures in Figure(2) using the $< DIV >$ tags. The ID attribute of each $< DIV >$ tag is used when we reference a particular $< DIV >$ tag in order to dynamically implement changes to it.

The $< DIV >$ tags of these data structures will be referred to as blocks. As mentioned in Section 3., only elements which are located within a block will be visible. As illustrated in Figure 6, all the blocks' *position* properties within the $STYLE$ attribute are given the same value. This means that all the blocks overlap on top of each other. However, only one data struture can appear at a time and we acheive this by manipulating each block's *visibility* property.

As explained in the previous Section, each layer is assigned with a key frame and appropriate metadata. We implement this feature in TV2Web by using nested layers within each block. Each nested layer within a particular block is referred to as a cell. A portion of the HTML source code for a particular block will look like the following (the $STYLE$ attribute has been omitted for clarity):

```
<DIV ID="block2>
  <DIV ID="cell21">
     <IMG ID="image21><br>
     ......................
  </DIV>
   <DIV ID="cell22">
     <IMG ID="image22><br>
     ......................
  </DIV>
</DIV>
```

The dotted lines in the code above represent the metadata of each cell.

5.3 Implementation of Basic Functions for Web Contents

In order to implement the browsing mechanism introduced in Section(4.), we use Dynamic HTML (DHTML) to incooperate animations into the blocks we created. A tutorial on DHTML is provided in [1]. After a particular data structure has zoomed beyond a certain threshold, we replace it by changing it's *visibility* attribute to *hidden*. A new layer appears by changing it's *visibility* attribute to *visible*.

5.3.1 Zoom-Out Function

The zoom-out function is used to gradually decreases the size of the image data being displayed. This function also gradually increases the metadata's level of details (LOD).

Figure(8) shows the concept behind the Zoom-Out function. We implemented the Zoom-Out function using JavaScript. A portion of the source code is shown below.

```
1.function zoomOut() {
2.  var i;
3.  if((image00.width>720 && image00.height>480)) {
4.      for(i=0;i<totalBlocks;i++){
5.        eval('block'+i+'.visibility="hidden"');
6.      }
7.      block0.visibility="visible";
8.      newHeight=newHeight-dy;
9.      newWidth=newWidth-dx;
10.     for(i=0;i<totalCells;i++){
11.     eval('image0'+i+'.width=newWidth');
12. eval('image0'+i+'.height=newHeight');
13.      }
14.      setTimeout("zoomOut()",30)
15.  }
.............................
```

Line 1. declares the Zoom-Out function. The **if** statement in line 3. makes sure that the loop loop is excuted only if the dimensions of images of a certain block are within a certain range. Line 7. changes the block's *visibility* property to *visible*. Lines 8. and 9. increment the image's dimensions by a constant amount (dy for the image's height and dx for the image's width). Line 10. runs a **for** loop for the total number of cells within a certain block. Lines 11. and 12 dynamically alter the width and height of the block's images within the html body. In line 14, if the Zoom-Out function is still being carried out, it's repeated recursively after a delay lasting 30 miliseconds. Once the dimensions of the image being viewed decreases beyond the bounding thresholds in line 3, the *visibility* property is set set to *hidden*.

5.3.2 Zoom-In Function

The Zoom-In function is used to gradually increase the size of the image data being displayed. This function also gradually decreases the metadata's level of details (LOD).

Figure(9) shows the concept behind the Zoom-In function. We implemented the Zoom-In function using JavaScript. A portion of the source code is shown below.

```
1.function zoomIn() {
2.  var i;
3.  if((image00.width<720 && image00.height<480)) {
4.      for(i=0;i<totalBlocks;i++){
5.        eval('block'+i+'.visibility="hidden"');
6.      }
7.      block0.visibility="visible";
8.      newHeight=newHeight+dy;
9.      newWidth=newWidth+dx;
10.     for(i=0;i<totalCells;i++){
11.     eval('image0'+i+'.width=newWidth');
12. eval('image0'+i+'.height=newHeight');
13.      }
14.      setTimeout("zoomIn()",30)
15.  }
.............................
```

Line 1. declares the Zoom-In function. The **if** statement in line 3. makes sure that the loop loop is excuted only if the dimensions of images of a certain block are within a certain range. Line 7. changes the block's *visibility* property to *visible*. Lines 8. and 9. increment the image's dimensions by a constant amount (dy for the image's height and dx for the image's width). Line 10. runs a **for** loop for the total number of cells within a certain block. Lines 11. and 12 dynamically alter the width and height of the block's images within the html body. In line 14, if the Zoom-In function is still being carried out, it's repeated recursively after a delay lasting 30 miliseconds. Once the dimensions of the image being viewed increases beyond the bounding thresholds in line 3, the *visibility* property is set set to *hidden*.

5.3.3 Focus Function

While viewing the web contents, especially when the textual LOD is increased, it might be impossible to view all the contents on the same display. On a tipical browser, this problem is solved by implementing scrollbars. In TV2Web,

we introduce the focus metaphor in order to overcome this problem. The focus function is implemented by pointing at a particular portion of the contents. This causes that portion to gradually move to the upper left portion of the display being viewed.

Initially, in Figure(10a), the Focus function is carried out on an entire cell. This causes that cell to move to the left most portion of the block. The corresponding cells move accordingly. In Figure(10b) a subregion within a cell is chosen. The Focus function will move that region to the the top left most portion of the block.

6. Conclusion

In this paper we automatically generate web contents after reconstructing the metadata in accordance to the video data's structure. We also propose a method to view (browse) the metadata together with the corresponding key frames in a browser. Furthermore, using the "zoom-in" and "zoom-out"metaphor, we introduce a method to seamlessly alternate between viewing video data and web contents. We also use the zoom metaphors to seamlessly alter the LOD of the contents being viewed.

The data models were implemented using Java Script and HTML $< DIV >$ tags. The zoom functions were implemented using DHTML. Furthermore, in order to replace scrollbars, we use the focus funtion in order to scroll through the contents.

Evaluation of our prototype and and support for multi stream video images remain as future works for this paper.

Acknowledgments

This research was partly supported by The Special Research Area's Grant In Aid For Scientific Research (2) For the Year 2002 under the project "Research for New Search Service Methods Based on The Web's Semantic Structure" (project no: 14019048, Representative: Katsumi Tanaka) and by The Scientific Research Fund Foundation (A)(2) for the year 2002 under the project "Multidata Searches And Views For Mobile Contents And The Generation Of Broadcasting Contents" (project no: 14208036, Representative: Katsumi Tanaka)

References

[1] The Dynamic Duo, Cross-Browser Dynamic HTML (available from http://www.dansteinman.com/dynduo/).

[2] HTML 4.01 Specification (available from http://www.w3.org/TR/html401/)

[3] Mahendren Munisamy, Sumiya Kazutoshi and Katsumi Tanaka : Automatic Transformation Of Video With Metadata Into Web Contents, IEICE Technical Report, Vol.102, No.208 DE2002 - 127, pp. 187 – 192, 18th July 2002.

[4] Jeromy Vineyard. Setting Up Your Shots. Micheal Wiese Productions, 1999.

[5] G. Ahanger and T.D.C Little. "A Survey of Technologies for Parsing and Indexing Digital Video", Journal of Visual Communication and Image Representation, 7(1):28-43, March 1996.

[6] Ricoh MovieTool available at (http://www.ricoh.co.jp/src/multimedia/MovieTool/)