Vision, Video, and Graphics (2003)
P. Hall, P. Willis (Editors)

Visualising Video Sequences Using Direct Volume Rendering

Gareth Daniel and Min Chen

Department of Computer Science, University of Wales Swansea, Swansea SA2 8PP, United Kingdom
csgareth@swansea.ac.uk, m.chen@swansea.ac.uk

Abstract

It is evident that more and more video data is being generated everyday, for example, by TV broadcast companies and security cameras. However, whilst we are overwhelmed by the huge amount of imagery data, machine vision is generally not yet ready to replace us in most of the everyday visual tasks. In this paper we present a novel approach to the handling of video data. We propose to employ volume visualisation techniques for "summarising" video sequences, and to render video volumes into appropriate visual representations. Such visualisations can be used to assist in a decision making process, for instance, to determine if there is any unusual level of activity recorded in a video. In the paper, we present a prototype system, called V^3, for capturing, managing, processing and visualising video data sets. We highlight the conceptual similarity between video visualisation and the traditional volume visualisation, and describe the deployment of conventional transfer functions and spatial transfer functions in video visualisation. We examine several statistical image comparison metrics and discuss their effectiveness in extracting meaningful information from video sequences. This work demonstrates the importance and the potential of combining graphics, video and vision technologies.

Keywords: video visualisation, video processing, volume rendering, image-swept volumes.

1. Introduction

The rapid advance of digital technologies has resulted in an explosion of digital imagery data. In particular, video data, generated by the entertainment industry, security and traffic cameras, video conferencing systems and not mentioning video emails, internet videos, etc., is perhaps most time-consuming to process. For example, an increasing problem in the security industry is the ratio of surveillance cameras to security personnel. It is simply not possible for any security officer to study a large number of video screens concurrently, while his/her attention can easily be drawn to a particular incoming video stream at any time. It is hence highly desirable to develop methods for extracting and highlighting interesting features in video sequences.

There is a rich collection of techniques for analysing imagery data, and for computing various statistical indicators. However, most of the techniques have not reached such an intelligent level that they can be relied upon to make decisions in place of a human. There is also a general lack of effective techniques to convey complex statistical information intuitively to a layperson such as a security officer.

In this paper, we present a novel approach to the handling of large volumes of video data. We propose to employ volume visualisation techniques for "summarising" video sequences, and to render video volumes into appropriate visual representations that can be used to assist in our decision processes. For example, when a security officer arrives at his/her office in the morning, he/she can be presented with one or a few visualisations for each surveillance camera that has been monitoring a premise during the previous night. From the visualisations, the officer can observe the level and patterns of activities recorded overnight, and decide if any specific section of a particular video needs to be replayed for further investigation. Video visualisation can also be used to assist in video processing, such a video segmentation.

The key to our approach is the volume visualisation technology, which has been successfully and extensively deployed in medical imaging and scientific visualisation. *Video data is a type of volume data*. Many statistical indicators of video data can also be represented in a volumetric form. This conceptual similarity allows us to utilise, in our work, some powerful volume visualisation and volume graphics techniques, such as *opacity transfer functions* and *spatial trans-*

fer functions. The results of this work have demonstrated the importance and the potential of combining graphics, video and vision technologies.

Our paper is organised as follows. In Section 2, we will briefly review the existing work on video processing and volume visualisation. In Section 3, we will describe the design and development of a prototype system, called V^3 (short for Volume Visualisation for Videos), which offers a system architecture for bringing together the technologies of video processing and volume visualisation. In Section 4, we will describe the use of volume modelling and rendering techniques for video visualisation. This will be followed by a discussion in Section 5 on several statistical image comparison metrics for extracting the "difference" features from video sequences. In Section 6, we will present some visualisation results and discuss the visual features in the visualisations. We will offer our concluding remarks and an indication of future work in Section 7.

2. Related Work

In 1997, Yeo and Yeung pointed out the needs for visualising video in order to "overcome the sequential and time-consuming process of viewing video" [9]. They suggested to use browsing techniques for viewing a video like flipping through a book. In recent years, a number of video database management systems were proposed and developed, all of which were focused on video archiving, segmentation, and contents management.

Many algorithms, which have been developed for processing images, can find their roles in video processing. Perhaps the most extensive use of such algorithms is in the areas of change detection [2], and content-based video retrieval [7].

During the past fifteen years, we have witnessed significant advances in volume visualisation and volume graphics [3], driven mainly by applications such as medical imaging and scientific computation. The work in this area has produced a large collection of methods that enable 3D information in a volumetric dataset to be selectively rendered into a single 2D image. The previous developments that relates strongly to this work includes direct volume rendering [6], constructive volume geometry [3] and image-swept volumes [8].

However, despite of the similarity between video data and volume data, there has not been much effort to bring video processing and volume visualisation together, perhaps except a demo by the Microsoft Research [5], and some attempts to introduce image comparison to visualisation [10].

3. V^3: System Overview

V^3 – *Volume Visualisation for Videos* – is a system designed to integrate a collection of techniques for capturing, managing, processing and visualising videos. It contains many utilities that can be effectively used for handling pre-recorded videos, for applications such as video segmentation. However, its primary design objective is to facilitate quick analysis of recently archived video data, such as in the security industry, through the use of volume visualisation. This objective is reflected strongly in the design of the V^3 system architecture and its user interface.

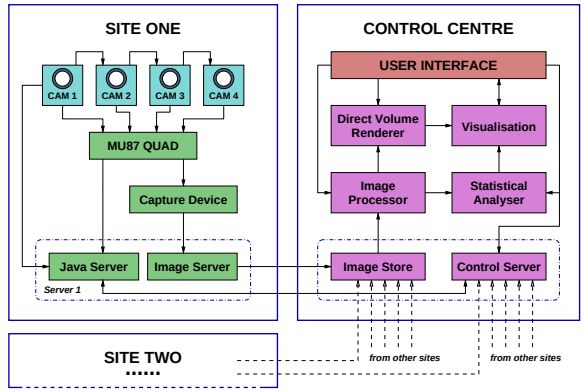


Figure 1: *System architecture of V^3.*

Figure 1 illustrates the overall system architecture of V^3, which allows multiple sites to be monitored concurrently in real-time from a single control centre. At each remote site, we have a set of cameras that can be interactively controlled (e.g., EVI-D31/B Sony video cameras). The imagery data captured by the cameras are combined by, and transported through, an MV87 Quad box, with which an individual or combined view can be selected at the control centre. The main software system of V^3 is expected to be installed in a control centre, where users can interactively control the remote cameras, select views, setting up recording processes, and most importantly "visualising" the captured data in many forms.

There are three major algorithmic modules in the software framework of V^3, namely *image processor*, *statistical analyser* and *volume renderer*. The image processor module consists of a number of image processing filters, change detection filters and image comparison metrics. The module takes the raw imagery data as inputs, and generates appropriate outputs for the statistical analyser and volume renderer modules. The statistical analyser takes inputs from the image processor module and produces numerical statistical indicators, which are then forwarded to the visualisation module where the statistical indicators are presented as 2D charts (such as line graphs, pie charts and bar charts). The actual functional boundary between the statistical analyser and image processor is a bit blurred in our implementation, because it is often more efficient to compute some basic statistics, such as mean pixel intensity of an image, during image processing. In general, statistical indicators local to an image are usually computed in the image processor module, while the global statistical indicators for the entire video, or any of

its sections, are computed in the statistical analyser module. The volume renderer module handles only volumetric data, which includes the raw video data as well as that generated by the image processor. One example of generated data is a sequence of difference images resulting from an image comparison metric. This modular design gives us the flexibility to replace existing metrics, filters and algorithms, and add new ones, whenever necessary.

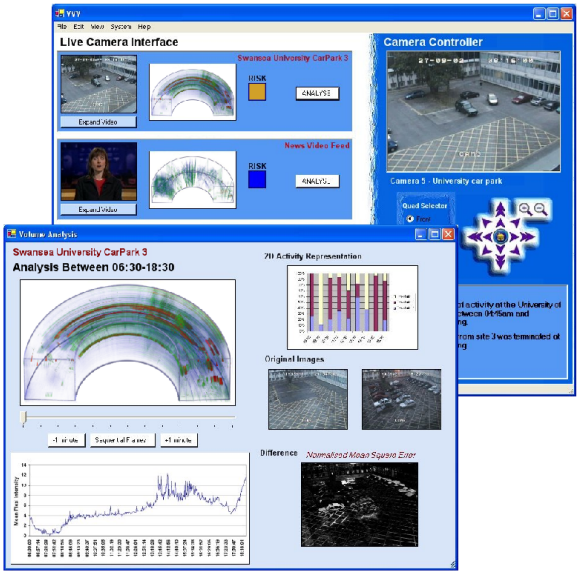


Figure 2: *The main screen layout of V*^{3}.

Figure 2 shows the two most important windows of V^3. The window at the back is the top-level window that shows a list of incoming video streams, together with the camera control facility for a selected site. The window in the front is a visualisation window for the video sequence from the selected site. It contains a sub-window for displaying a rendered visualisation image and another for a statistical chart. In addition, it offers displaying areas for selecting and managing image processing filters and comparison metrics.

The Microsoft Visual C# .NET development environment has been used to implement the main software components of V^3, though many filters, metrics and algorithms were first implemented in C and tested in a Linux environment.

4. Rendering Video Datasets

A video data set V is composed of a series of images $I_1, I_2, \ldots, I_{tn}$, where all images are normally of the same resolution $xn \times yn$. Hence V can be considered as a collection of *voxels* that are organised into a 3D regular grid as:

$$V = \{v_{x,y,t} | 1 \leq x \leq xn, 1 \leq y \leq yn, 1 \leq t \leq tn\}.$$

Each voxel v is addressed by its grid coordinates (x, y, t), and is associated with one or more scalar values representing imagery properties such as intensity and colour components. In *volume visualisation*, such a structure is commonly referred to as a *volume data set*, *3D raster* or a *volume buffer*. Because the t dimension is of a different nature from that of the x or y dimension, V should normally be manipulated as an anisotropic grid, whenever the spacing between neighbouring voxels is a matter of interest.

The principal objective of volume visualisation is to extract meaningful information from volumetric data using computer graphics. Volume rendering techniques, which have been extensively deployed in medical imaging and scientific visualisation, allow information contained in a volume data set to be selectively rendered into a single 2D image. This easily leads to the desire for visualising information contained in a video data set.

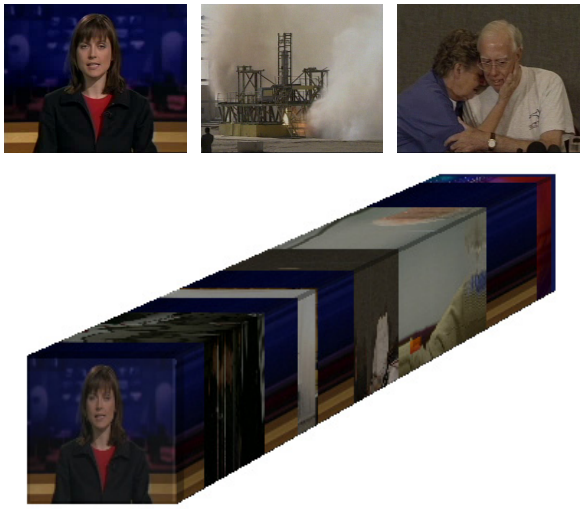
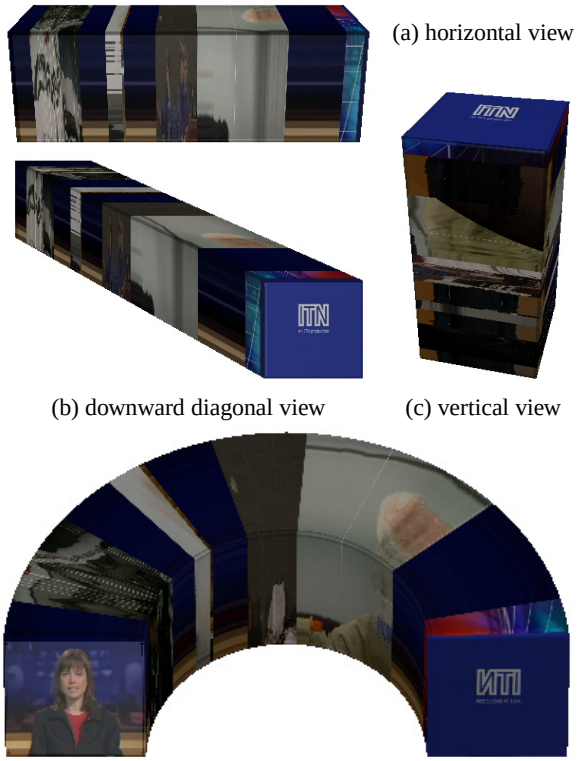


Figure 3: *A video data set is a spatial object.*

Like conventional volume data sets, when coupled with an interpolation function, such as trilinear interpolation, a video data set V is essentially a *spatial object* $\mathbf{o}$, which is composed of a set of geometrically-bounded attribute fields $(A_0, A_1, \ldots, A_k)$. Let $\mathbb{R}$ denote the set of all real numbers, and $\mathbb{E}^3$ denote 3D Euclidean space. Each attribute field is a scalar field function $F : \mathbb{E}^3 \rightarrow \mathbb{R}$. A typical raw RGB video data set is thus a discrete specification of a spatial object with three attribute fields, namely red, green and blue channels. In Figure 3, an 80-second sequence of a recorded television news programme, for which three example frames were shown, is treated as a spatial object, and displayed as a volume of colour points.

When information contained in a 3D spatial object is extracted and rendered into a 2D image, it is inevitable that some visual features may be obscured by others. This problem can normally be dealt with through interactive manipulation of the camera parameters such as the viewing position.



(a) horizontal view

(b) downward diagonal view

(c) vertical view

(d) horseshoe view

Figure 4: *In addition to the upward diagonal view shown in Figure 3, V^3 provides another four different views.*

However, this is not always practical to a security officer who looks after several dozens of cameras. Hence one of our objectives is to provide users with some intuitive but powerful visual representations in order to facilitate a quick decision-making process. We experimented with many designs and selected five representations as standard options in V^3. In addition to the upward diagonal view shown in Figure 3, V^3 also provides a horizontal view, a vertical view, a downward diagonal view, and a horseshoe view (Figure 4).

In general, the horseshoe view conveys more information than the other four views, though it shows a horizontally-flipped image at its right end. The construction of such a visual representation is achieved by employing the *image-swept volume* technique [8]. Instead of deforming a video volume directly during modelling, we associate the object with a *spatial transfer function*, $\Psi : \mathbb{E}^3 \rightarrow \mathbb{E}^3$. Ψ defines the geometrical transformation of every point p in $\mathbb{E}^3$. It is used to modify the sampling position of a scalar field A during rendering in the form of $A'(p) = A(\Psi(p))$. Direct rendering of a spatial object using a ray casting algorithm [6] is essentially a discrete sampling process for evaluating scalar fields associated with the spatial object. With Ψ, an evaluation of A' at p implies the evaluation of A at $q = \Psi(p)$.

Chen *et al* [4] recently demonstrated that spatial transfer functions can be defined as spatial objects, and they can be integrated into a scene graph in the same way as conventional spatial objects. In V^3, we have a built-in scene graph that includes a spatial transfer function node, which is only activated for the horseshoe view. The spatial transfer function $q = \Psi(p)$ is a semi-circular sweep. Consider our video volume is defined in a normalised coordinate system of the domain $[0,1]^3$. Let $r = \sqrt{(p_x - 0.5)^2 + (p_y - 0.5)^2 + (p_z - 0.5)^2}$ and $\phi = arctan(p_z - 0.5, p_x - 0.5) \in [-\pi, \pi]$. We have:

$$q_x = \begin{cases} 2 - 4r & 0.25 \leq r \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$q_y = 1 - p_y$$

$$q_z = \begin{cases} 1 - \phi \div \pi & \phi \geq 0 \\ 0 & \phi < 0 \end{cases}$$

Opacity and colour transfer functions (which are often referred to simply as transfer functions) are an intrinsic part of volume visualisation, and in particular, direct volume rendering. It is common to define an attribute field upon another using a transfer function, usually in the form of $A_i(p) = \Phi(A_j(p)), p \in \mathbb{E}^3$, where $\Phi : \mathbb{R} \rightarrow \mathbb{R}$. During rendering, transfer functions are used to select what information is to be visualised by modifying the opacity field of a spatial object, or determine how information is to be displayed by modifying its colour fields.

In V^3, each object $\mathbf{o} = (O, R, G, B, D)$ is defined with five attribute fields, namely opacity, red, green, blue and data. For example, the spatial object **o** shown in Figure 3 is in fact associated with a uniform, fully opaque, opacity field within the bounding volume. The data field D is usually used to represent non-visual data, such as a difference volume, and facilitates the normal estimation. In the following discussions, we assume that the values of all these five scalar fields are normalised to the domain $[0, 1]$. (It is common in image processing to denote an opacity channel using α or A. In volume graphics and field-based modelling, normally the same alphabet is applied consistently to all fields including opacity, colour components, normal, reflection, etc. Both conventions can be adopted in this paper, as we do not include many other fields in our discussions. However, we feel it is more appropriate to use $O(p)$ for the opacity, emphasising the fact that it is a 3D scalar field and is one of the attribute fields of a spatial object.)

Let us construct a visualisation by defining a non-uniform opacity field O based upon the hue property of the spatial objects. From the RGB fields of **o**, we first obtain HSV components, hue $H(p) \in [0, 360)$, saturation $S(p) \in [0, 1]$, value $V(p) \in [0, 1]$. We then define the opacity field as:

$$O(p) = \begin{cases} 0.2V(p) & 225 \leq H(p) \leq 255 \\ 1 & \text{otherwise} \end{cases}$$

This transfer function results in the visualisation shown in

Figure 5, which turns parts of the objects with blue as the dominant wavelength, such as the blue background behind the newscaster, into translucent amorphous matters.

Figure 5: *The application of a transfer function.*

Many features of a video sequence can also be represented by a volume data set. For example, we can construct a volume data set that represents the relative difference between consecutive images in V, that is:

$$\Delta(I_1, I_2), \Delta(I_2, I_3), \ldots, \Delta(I_{tn-1}, I_{tn}) \quad (1)$$

where Δ is a difference function that operates on images. Here we simply assume that $\Delta(I_i, I_{i+1})$ results in an image representing some form of visual difference between I_i and I_{i+1}. The sequence of difference images is in itself a volume data set, and can be used to assist in the visualisation of the original video data set. In the next section, the computational specification of such a difference function will be considered. The use of difference images in visualisation will be examined in Section 6.

5. Computing Image Difference

In this section we will consider several different image comparison metrics. We concentrate on the global statistical metrics which allow us to compare, in Section 6, the effectiveness of visualisations and statistical indicators. We consider the capability of each metric for highlighting the geometrical difference caused by moving objects while de-highlighting any luminance and colour difference caused by the change of lighting conditions.

There are various colour spaces where image comparison may take place. In this paper, we focus on the YIQ space, which is a linear transformation of the RGB space:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The main advantage of using the YIQ model is that the luminance information is decoupled from the colour information, and this allows us to examine the effectiveness of difference computation in terms of luminance and colour separately. Although it is desirable to employ a perceptually uniform colour space to compute colour difference, this would make V^3 dependent on individual colour matching specifications for different video capture devices.

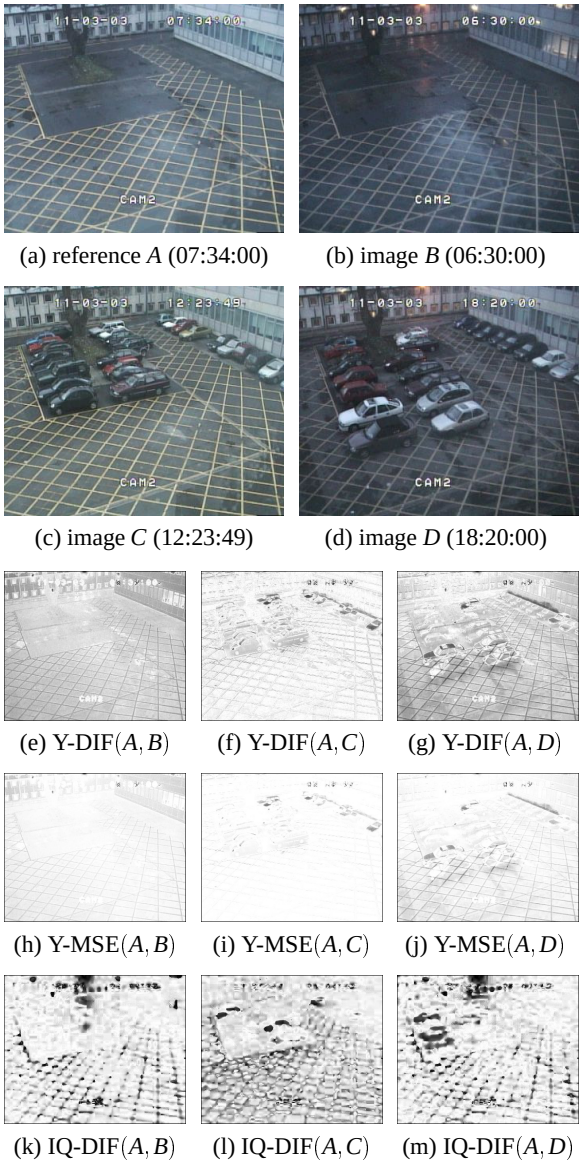
Let us consider first three comparison metrics, namely Y-DIF, Y-MSE and IQ-DIF.

- Y-DIF(I_1, I_2) – *simple difference metric* – It takes two input images, I_1 and I_2, and computes a grey-scale output image O where each pixel represents the linear distance between the Y-values of two corresponding pixels in I_1 and I_2 respectively.
- Y-MSE(I_1, I_2) – *mean squared error* metric – Instead of the linear distance, it computes the squared distance (i.e., error) between the Y values of each pair of corresponding pixels. The name of the metric is inherited from the corresponding statistical indicator that calculates the mean of the squared errors of all pixels in an image.
- IQ-DIF(I_1, I_2) – *colour difference metric* – It computes the angle between the IQ vectors of the two corresponding pixels in I_1 and I_2, and sets the corresponding pixel value in O to the angle. It gives a result similar to that obtained by computing the hue difference in the HSV space.

Figure 6 shows a reference image A, and a set of three example images, B, C and D, which are compared against the reference image A. All images were extracted from a surveillance video of a university car park. The reference image A shows an empty car park in a reasonably good lighting condition. B, C and D represent images that exhibit different levels of activities and are taken in different lighting conditions. The difference images computed using Y-DIF, Y-MSE and IQ-DIF are also shown in Figure 6. In order to maintain a consistent evaluation, we scale the value range of each output image from its individual min-max range to [0, 255]. As this scaling process is image-dependent, it is not suitable for the general use in V^3.

From Figure 6, we can see that IQ-DIF does not perform as well as what one would expect. This is partially due to the fact that all images were JPEG-compressed by the image capturing device. The compression seems to be optimised for luminance at the cost of redistributing colours within small regions across the image. Y-DIF seems to be affected badly by the lighting conditions, while both Y-DIF and Y-MSE have some difficulties to distinguish geometrical difference from luminance difference. This naturally leads to the process for normalising image luminance.

Metrics Y-NDIF and Y-NMSE are the normalised version of Y-DIF and Y-MSE respectively. Before we apply Y-DIF(I_1, I_2) and Y-MSE(I_1, I_2), we first normalise the Y-component of each input image based on its mean value and standard deviation. To a certain extent, this may reduce the luminance difference caused by different lighting conditions. Ideally one could carefully select a "geometrically-static" section in the images for guiding the normalisation. In practice, it is not always feasible. In our example, the large



(a) reference A (07:34:00) (b) image B (06:30:00)

(c) image C (12:23:49) (d) image D (18:20:00)

(e) Y-DIF(A,B) (f) Y-DIF(A,C) (g) Y-DIF(A,D)

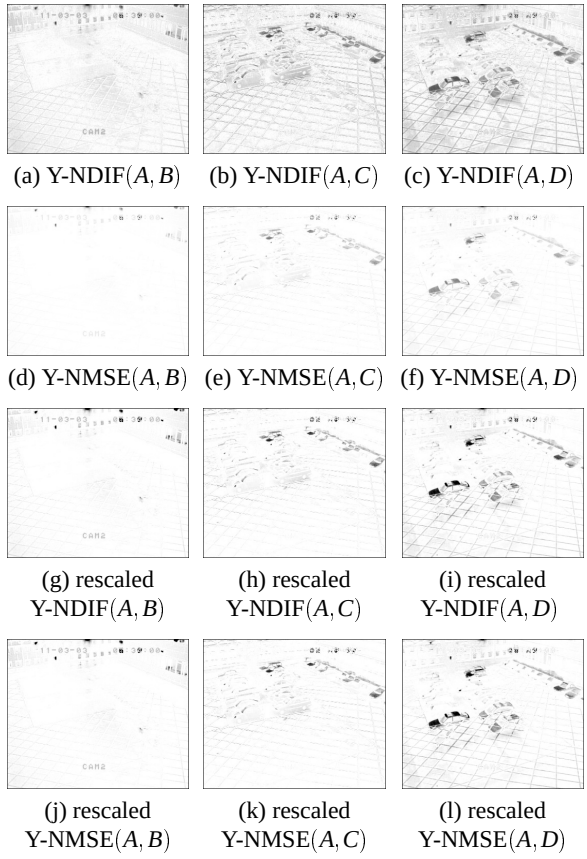
(h) Y-MSE(A,B) (i) Y-MSE(A,C) (j) Y-MSE(A,D)

(k) IQ-DIF(A,B) (l) IQ-DIF(A,C) (m) IQ-DIF(A,D)

Figure 6: *Three captured images in (b), (c) and (d) are compared with a reference image in (a) using three different metrics, namely Y-DIF, Y-MSE and IQ-DIF. The value ranges of the resultant images are re-mapped to the [0, 255] domain for comparative evaluation. In addition, all difference images, (e) - (m), have been inverted for clearer printing.*

section covering the ground is "geometrically-dynamic", due to the movement of cars. Although the section covering the building is relatively geometrically-static, its luminance does not change uniformly, as windows are affected by individual office lights that are switched on or off in an unpredictable manner.

Figure 7 shows the results of applying Y-NDIF and Y-NMSE to the same set of examples in Figure 6. To help visualising the three images in each row in a consistent manner, we also rescale the original results through multiplying by a constant, i.e., 30 for Y-NDIF and 10 for Y-NMSE. The level of activity in general is better conveyed in those images.



(a) Y-NDIF(A,B) (b) Y-NDIF(A,C) (c) Y-NDIF(A,D)

(d) Y-NMSE(A,B) (e) Y-NMSE(A,C) (f) Y-NMSE(A,D)

(g) rescaled Y-NDIF(A,B) (h) rescaled Y-NDIF(A,C) (i) rescaled Y-NDIF(A,D)

(j) rescaled Y-NMSE(A,B) (k) rescaled Y-NMSE(A,C) (l) rescaled Y-NMSE(A,D)

Figure 7: *With the Y-NDIF metric, normalisation was applied prior to the difference operation, resulting in (a), (b) and (c). Similarly Y-NMSE results (d), (e) and (f). In (a)-(f), the value ranges of the resultant images are re-mapped to the [0, 255] domain for comparative evaluation. We can usually apply a constant scaling factor for a sequence of images, and this effectively acts as a transfer function for direct volume rendering. In (g), (h) and (i), the original results of Y-NDIF are rescaled by a factor of 30, and in (k), (l) and (m), those of Y-NMSE, are rescaled by a factor of 10. All images in this figure have been inverted for clearer printing.*

6. Results and Remarks

A sequence of difference images is also a volume data set, and can thereby be visualised using volume rendering techniques. We may render such a volume data set to highlight some statistical features of the original video data set. Our first example is to examine the effectiveness of using video

visualisation for identifying the transition frames between different segments of a news video, with a particular focus on the correlation between visualisation and statistical indicators. We experimented with four metrics, Y-DIF, Y-MSE, Y-NDIF and Y-NMSE, for computing difference volumes, from which visualisations are obtained using appropriate opacity and colour transfer functions.

Two of such visualisations, associated with Y-DIF and Y-NMSE, are shown in Figure 8, together with line graphs (in Figure 9) which depict the mean intensity of each difference image computed using both Y-NMSE and Y-DIF, and the manually identified transition points. The intensity of the amorphous matters in the visualisation represents the changes between consecutive images. In some cases the original image patterns are visible, and in some other cases the image frames can be identified, indicating a major change between two different segments. As shown in the figure, the visualisation associated with Y-DIF conveys visual information that is consistent with the statistics shown in the corresponding line graph, though Y-DIF misinterprets many camera flashes (15:00:07-15:00:16, 15:00:51) as segment transition points. The visualisation associated with Y-NMSE (with a scaling factor of 10) is less effective in highlighting transition frames, though the corresponding statistical indicators are more consistent with manually identified transition frames.

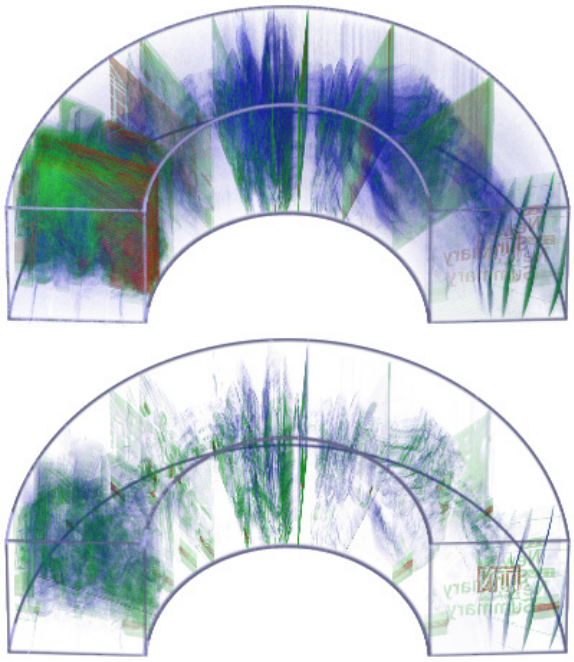


Figure 8: *Visualisation of transition frames in an ITN news video. Y-DIF (top) and Y-NMSE (bottom) are used to compute the difference volumes respectively.*

Our second example involves the car park video sequence,

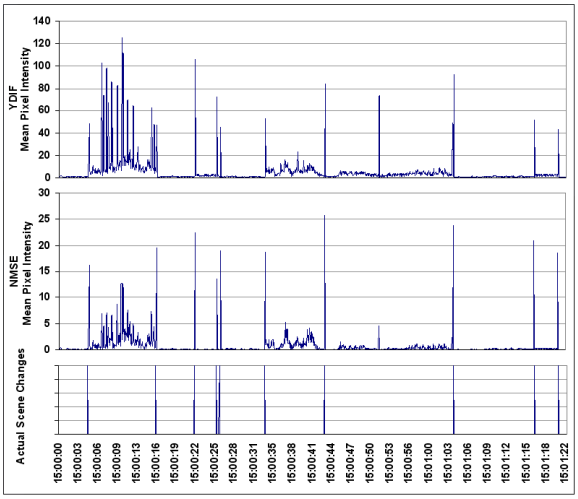


Figure 9: *Statistical indicators corresponding to Figure 8. Top: mean intensity of difference images computed using Y-DIF. Middle: using Y-NMSE. Bottom: manually identified transition frames.*

which contains 662 images taken over a 12 hour period. The visualisation in Figure 10 depicts the relative difference between consecutive images in the sequence as defined in Eq(1). On the other hand, the visualisation in Figure 11 shows the absolute difference between each image in the sequence and a reference image R, that is:

$$\Delta(I_1, R), \Delta(I_2, R), \ldots, \Delta(I_{tn}, R)$$

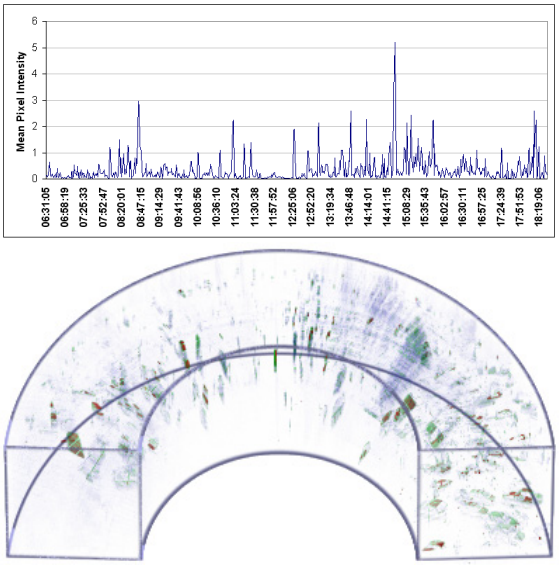


Figure 10: *The visualisation of a "relative" difference volume computed from the car park video sequence.*

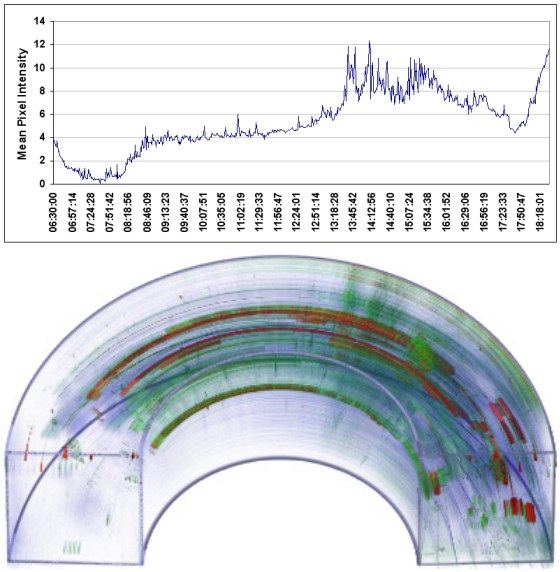


Figure 11: *The visualisation of an "absolute" difference volume computed from the car park video sequence.*

Both difference volumes are visualised on their own with a colour transfer function, indicating the scale of changes (i.e., red for large intensity changes, green for medium and blue for small). Figures 10 offers a visual representation indicating the level of activities during the recording period, that is, movement of cars. The same pattern of activities are shown in the visualisation and the line graph. Figure 11 gives an interesting visualisation, where the swept lines indicate many stationary cars in a large part of the recording period. The visualisation shows the level of usage of the car park, with little occupancy in the early morning, a full car park during the day, and some dynamic activities in the evening when staff were leaving for home and evening students were coming to the university. On the video, there was a major change of the weather condition during the afternoon, and this change is clearly visible from the line graph which could misinform us of some extra activities or occupancy. However, in the visualisation, it is much easier to discard such changes as the amorphous blue patterns are perpendicular to the line time.

7. Conclusions and Future Directions

We have described an approach that can effectively "summarise" a video sequence and can be deployed to deal with the problem of the rapid explosion of video data. We have shown that video data can be processed and visualised in the same manner as other volumetric data. We have examined several statistical image comparison metrics. With the aid of two example video data sets, a television news programme and a surveillance video, we have demonstrated the usefulness of video visualisation. In many cases, visual representations of a video conveys more information than statistical results presented in graphs. Among those metrics considered, we have found that Y-NMSE correlates well with statistical indicators. Y-DIF can result in good quality visualisation for videos captured in well-controlled conditions, but may easily lead to misrepresentation in situations where luminance varies dynamically.

Our future work will have two strands, (i) the continuing investigation into image comparison metrics in order to improve the effectiveness of the difference calculation, and (ii) the development of a progressive volume rendering algorithm for rendering images when they are captured.

References

1. P. Alshuth, T. Hermes, L. Voigt and O. Herzog, "On video retrieval: content analysis by ImageMiner^{TM}". *Proc. SPIE: Storage and Retrieval for Image and Video Databases*, pp. 236–249, (1998).
2. M. Brocke, "Statistical image sequence processing for temporal change detection", *Proc. 24th DAGM Symposium: Pattern Recognition*, Zurich, Switzerland, LNCS 2449, pp. 215–223, (2002).
3. M. Chen, "Volume graphics", in A. Kent and J.G. Williams (eds.) *Encyclopedia of Microcomputers*, **26**, pp. 363–387, Marcel Dekker, New York, (2001).
4. M. Chen, D. Silver, A. S. Winter, V. Singh and N. Cornea, "Spatial transfer functions — a unified approach to specifying deformation in volume modeling and animation", to appear in *Proc. Volume Graphics 2003*, Japan, (2003).
5. A. W. Klein, P. J. Sloan, A. Finkelstein and M. F. Cohen, "Stylized video cubes", *Proc. the ACM SIGGRAPH Symposium on Computer Animation*, pp. 15–22, (2002).
6. M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, **8**(5), pp. 29–37, (1988).
7. J.-Y. Pan and C. Faloutsos, " VideoCube: a novel tool for video mining and classification", *Proc. the Fifth International Conference on Asian Digital Libraries (ICADL 2002)* pp. 11–14, (2002).
8. A. S. Winter and M. Chen, "Image-swept volumes", *Computer Graphics Forum*, **21**(3), pp. 441–450+640, (2002).
9. B. Yeo and M. M. Yeung, "Retrieving and visualizing video", *Communications of the ACM*, **40**(12), pp. 43–52, (1997).
10. H. Zhou, M. Chen and M. F. Webster, "Comparative evaluation of visualization and experimental results using image comparison metrics" *Proc. IEEE Visualization 2002*, pp. 315–322, Boston, (2002).