Compression of Temporal Video Data by Catmull-Rom Spline and Quadratic Bézier Curve Fitting

Murtaza Khan
Graduate School of Science & Technology
Keio University, Japan
murtaza@on.ics.keio.ac.jp

Yoshio Ohno
Faculty of Science & Technology
Keio University, Japan
ohno@on.ics.keio.ac.jp

ABSTRACT

This paper presents a new method for lossy compression of temporal data of both naturally recorded and synthetically created videos by Catmull-Rom spline and quadratic Bézier curve fitting. The proposed method approximates the luminance or color variations in a sequence of frames by spline fitting in Euclidean space. Precise control of accuracy at pixel level is achieved by a specified tolerance of error. A break and fit criterion is employed to minimize the number of curve segments required to fit the data. Experimental results show that the described method yields very good results, both in terms of objective and subjective quality measurement, i.e., bit-rate/PSNR and human visual acceptance, without causing any blocking artifacts.

Keywords: Video data, sequence of images, approximation, compression, fitting, Catmull-Rom spline, Bézier curve.

1 INTRODUCTION

Digital video data consists of sequence of frames (images). Each frame consists of rectangular 2-D array of pixels. 3-D *RGB* or 1-D *luminance* values in a sequence of frames are associated with each pixel. *RGB* or *luminance* value(s) of a pixel can be considered as a point in Euclidean space R^3 or R^1 respectively. Let a video consists of a sequence of n frames. Frame width and height are W and H respectively. Then for each spatial location (x_i, y_j), $1 \leq i \leq W$, $1 \leq j \leq H$, we have temporal video data in n frames, $\{p_1, p_2, \ldots, p_n\}$, i.e., $p_j = (R_j, G_j, B_j)$ for *RGB* or $p_j = I_j$ for *luminance*, where $1 \leq j \leq n$. Figure 1 shows *RGB* variation of a pixel in 80 frames of a video. Video data contains temporal and spatial correlation. In our proposed method focus is on temporal compression of video data by approximating it using quadratic Bézier curve and Catmull-Rom spline fitting.

Splines and curves are widely used in computer-aided design and computer graphics because of the simplicity of their construction, accuracy of evaluation, and their capability to approximate complex shapes through curve fitting and interactive curve design [BBB95]. Spline can compress the data by approximating large number of data points with far less number of control points. Control points can be encoded by some appropriate encoding technique. During the decoding process approximated data points are generated by spline interpolation of control points.

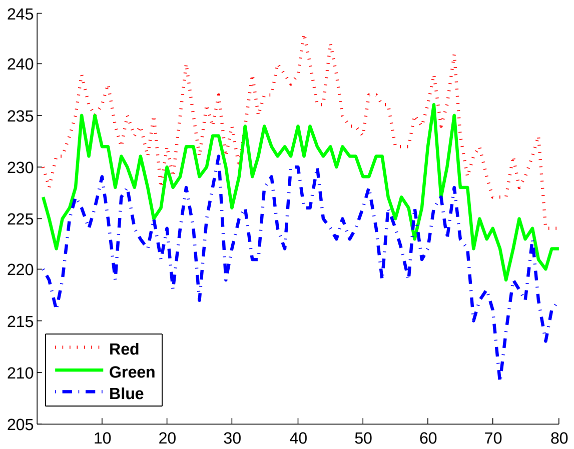

Figure 1: *RGB* temporal variation of a pixel in 80 frames of a video.

In our method, we considered temporal variations of color or luminance values of pixels in a sequence of frames as input data and approximated it with far less number of control points (output data). An important factor in spline approximation of data is finding least number of control points. We achieved this goal by selecting optimal set of control points.

Approximation and compression of data using parametric curves particularly cubic splines are explored by many authors [LCT07, UM00, IO93] et al. A method of dynamic mesh compression of animated sequences is described by [LV07]. The approach used by [LV07] is based on EdgeBreaker and Principal Component Analysis (PCA) and it exploits both spatial and temporal coherence. [LCT07] presented a medical image compression algorithm using cubic spline. Contour data compression method using Curvature Scale Space technique and Hermite curves is proposed by [UM00]. Proposed method of [IO93] uses cubic Bézier curves and is suitable for compressed representation of outline of

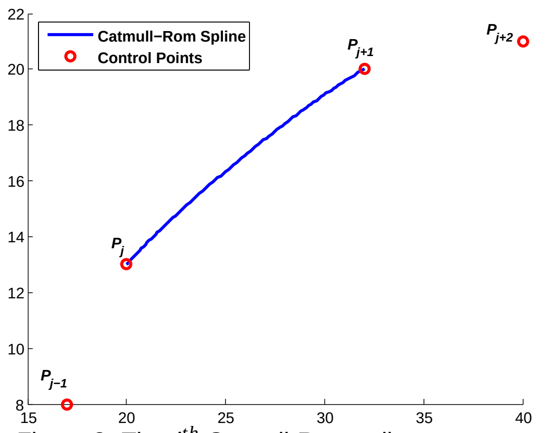

Figure 2: The j^{th} Catmull-Rom spline segment.

fonts. Our method uses quadratic Bézier curve and Catmull-Rom spline that are computationally more efficient than cubic splines. Cubic splines are more appropriate for image compression but less feasible for video data compression. Non-spline based temporal correlation reductions methods are based on motion estimation via translating block matching algorithms (BMAs) [CP02, SD02, KIH+81]. In a typical BMA, a frame is divided into rectangular blocks of pixels. Then the current (predicted) block is matched against blocks in the previous frame, for a maximum motion displacement of w pixels. The best match on the basis of a *mean absolute error* (*MAE*) criterion yields displacement relative to current block called motion vector. Predicted frame is approximated by blocks in reference frame and corresponding motion vectors [Gha03, Thy05, Say05]. In contrast to BMAs, we do not find matching pixel or matching block. We adopts different approach of fitting i.e., approximating the change in color or luminance values of each pixel in a sequence of frames, at the fixed spatial location (without translation of block/pixel), by quadratic Bézier curve and Catmull-Rom spline fitting. BMA works at block level and may cause blocking artifacts [WOZ01, IM00]. Due to pixel level fitting by proposed method, precise control of accuracy and immunity from blocking artifacts is achieved. Due to large size of video data it is also desirable that fitting process is automated. In our proposed scheme, the user has just to initialize a few parameters, then the rest of the steps i.e., fitting, encoding/decoding are fully automated.

2 CATMULL-ROM SPLINE AND QUADRATIC BÉZIER CURVE

The following two subsections i.e., 2.1 and 2.2 describe the theory and mathematical models of Catmull-Rom spline and quadratic Bézier curve, respectively.

2.1 Catmull-Rom Splines (CRS)

Catmull-Rom spline (CRS) [CR74] is a piecewise C^1 continuous curve with specified endpoint tangents. A CRS segment is defined by four control points, i.e., P_{j-1}, P_j, P_{j+1} and P_{j+2}, as shown in Figure 2. The j^{th} segment of CRS interpolates between two *middle control points*, i.e., P_j and P_{j+1}. The *end control points*, i.e., P_{j-1} and P_{j+2} are used to calculate the tangents of P_j and P_{j+1}. Equations for boundary conditions of j^{th} segment are written as:

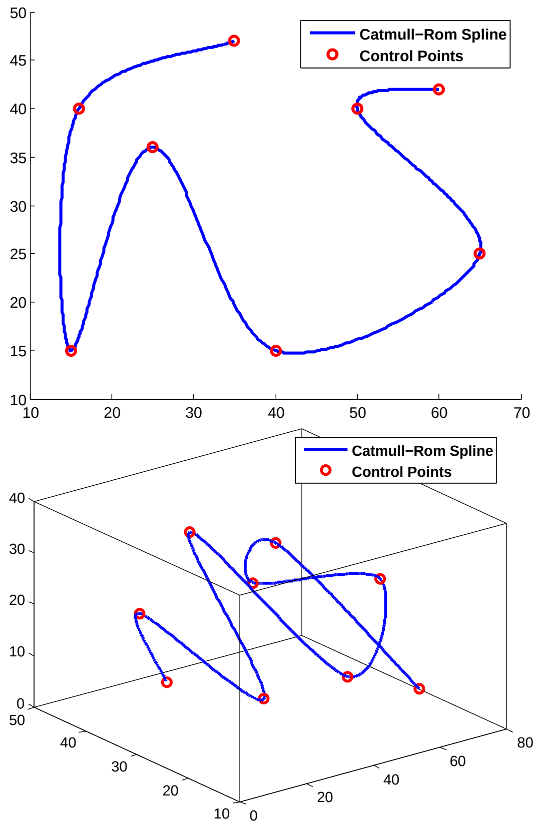

Figure 3: Multi-segment Catmull-Rom splines in 2-D and 3-D space.

$$P'_j = \frac{1}{2}\left(P_{j+1} - P_{j-1}\right), \quad (1)$$

$$P'_{j+1} = \frac{1}{2}\left(P_{j+2} - P_j\right). \quad (2)$$

For l joined segments, there are $2l$ conditions for continuity of functions and $2l$ conditions for continuity of slopes. Finally the equation of CRS for j^{th} segment is written as follows:

$$\begin{aligned} Q_j(t_i) =& \frac{1}{2}[(-t_i^3 + 2t_i^2 - t_i)P_{j-1} \\ &+ [3t_i^3 - 5t_i^2 + 2]P_j \\ &+ [-3t_i^3 + 4t_i^2 + t_i]P_{j+1} \\ &+ (-t_i^3 - t_i^2)P_{j+2}], \end{aligned} \quad (3)$$

where t_i is parameter of interpolation , $0 \le t_i \le 1$. In order to generate n points between P_j and P_{j+1} inclusive, the parameter t_i is divided into $(n-1)$ intervals

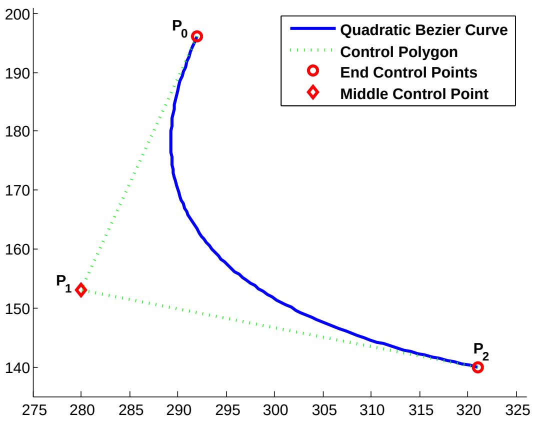

Figure 4: A quadratic Bézier curve segment.

between 0 and 1 inclusive, and $Q_j(t_i)$ is evaluated at n values of t_i. Since the CRS passes through its *middle control points*, therefore $Q_j(0) = P_j$ and $Q_j(1) = P_{j+1}$. Figure 3 shows multi-segment Catmull-Rom splines.

2.2 Quadratic Bézier Curve (QBC)

Quadratic Bézier curve (QBC) is a C^0 continuous curve. A QBC segment, is defined by three control points, i.e., P_0, P_1, and P_2, as shown in Figure 4. P_0 and P_2 are called *end control points* (*ECP*), while P_1 called a *middle control point* (*MCP*). QBC passes through its *end control points*, while a *middle control point* is used to control the shape of curve. To generate continuous quadratic Bézier curves that interpolate $k+1$ points k curve segments are used. Equation of a QBC segment can be written as follows:

$$Q(t_i) = (1-t_i)^2 P_0 + 2t_i(1-t_i)P_1 + t_i^2 P_2, \qquad (4)$$

where t_i is a parameter of interpolation , $0 \leq t_i \leq 1$. In order to generate n points between P_0 and P_2 inclusive, the parameter t_i is divided into $(n-1)$ intervals between 0 and 1 inclusive, and $Q(t_i)$ is evaluated at n values of t_i. Since the QBC passes through its first and last control points, therefore $Q(0) = P_0$ and $Q(1) = P_2$. Figure 5 shows multi-segment quadratic Bézier curves.

3 FITTING STRATEGY

This section describes the strategy of fitting Catmull-Rom spline and quadratic Bézier curve to video data. Fitting process is applied to temporal data of each spatial location individually. Number of spatial locations are $W \times H$, where W and H are width and height of a frame respectively. Let color or luminance value of a spatial location (x,y), $1 \leq x \leq W$, $1 \leq y \leq H$, at frame i is p_i, where $0 \leq p_i \leq 255$ and $1 \leq i \leq n$. We have to approximate the n values of each spatial location by quadratic Bézier curve and Catmull-Rom spline fitting. Now we describe the fitting process of an arbitrary spatial location (x_i,y_i). The temporal data of (x_i,y_i) in

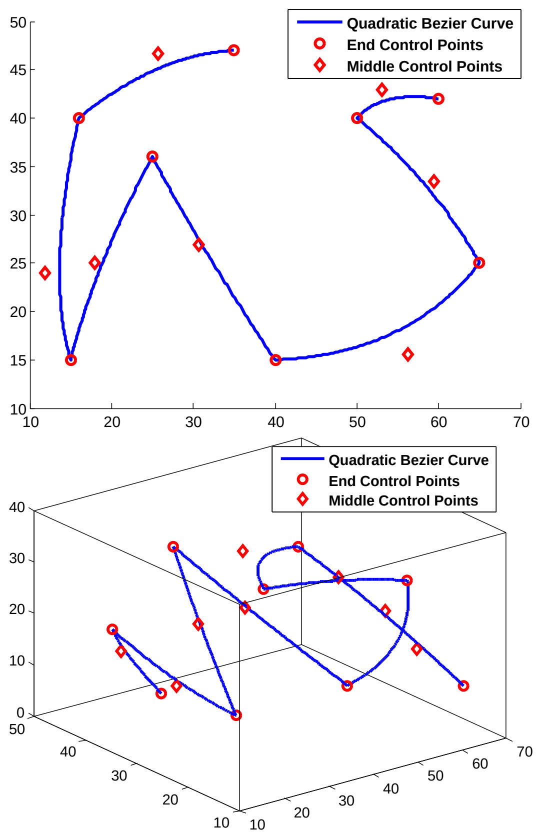

Figure 5: Multi-segment quadratic Bézier curves in 2-D and 3-D space.

n frames is $O = \{p_1, p_2, \ldots, p_n\}$. As an input to algorithm the user specifies two parameters: (1) *upper limit of error* ξ^{lmt}, i.e., maximum allowed squared distance between original and fitted data, e.g., $\xi^{lmt} = 100$, and (2) *initial breakpoint interval* δ, i.e., color or luminance value of pixel after every δ^{th} frames is taken as a breakpoint (control point), e.g., $\delta = 12$ sets the initial breakpoints as $BP = \{p_1, p_{13}, p_{25}, p_{37}, \ldots, p_n\}$ (color or luminance value of pixel in last frame is always taken as a breakpoint). The fitting process divides the data into segments based on breakpoints, i.e., $S = \{S_1, S_2, \ldots, S_{u-1}\}$. A segment is a set of all points (color or luminance values) between two consecutive breakpoints, e.g., $S_1 = \{p_1, p_2, \ldots, p_{13}\}$, $S_2 = \{p_{13}, p_{14}, \ldots, p_{25}\}$.

In addition to control points of the current segment, Catmull-Rom spline needs control points of previous and next segments. Therefore we used breakpoints of previous, current and next segments as control points of a current Catmull-Rom spline segment and obtained the approximated data of current segment using Eq. (3). An interesting question is how to obtain the breakpoints of first and last segments because they do not have previous and next segments respectively. Unlike [KB84] taking arbitrary breakpoints at both ends, we opted to take $P_{j-1} = P_j$, for the first segment and $P_{j+2} = P_{j+1}$, for the last segment. This way we are able to get the control points of all segments automatically. For a quadratic Bézier curve breakpoints of current segment are taken as *end control points* (*ECP*), while *middle control point* (*MCP*) i.e., P_1 is obtained by least square method. Least square method gives the *best* value of *middle control point* that minimizes the squared distance between the original and the fitted data. If there are m data points in a segment, and O_i and $Q(t_i)$ are values of original and approximated points respectively then we can write the least square equation as follows:

$$U = \sum_{i=1}^{m} [O_i - Q(t_i)]^2. \tag{5}$$

Substituting the value of $Q(t_i)$ from Eq. (4) in Eq. (5) yields:

$$U = \sum_{i=1}^{m} [p_i - (1-t_i)^2 P_0 + 2t_i(1-t_i)P_1 + t_i^2 P_2]^2. \tag{6}$$

To find value of P_1 differentiating Eq. (6) partially with respect to P_1 yields:

$$\frac{\partial U}{\partial P_1} = 0. \tag{7}$$

Solving Eq. (7) for P_1 gives:

$$P_1 = \frac{\sum_{i=1}^{m} \left[p_i - (1-t_i)^2 P_0 - t_i^2 P_2\right]}{\sum_{i=1}^{n} 2t_i(1-t_i)}. \tag{8}$$

Once all three control points are in hand then approximated data of current Bézier segment is obtained using Eq. (4).

Same procedure is repeated for each segment for both Catmull-Rom spline and quadratic Bézier curve. This yields n values of approximated data, $Q = \{q_1, q_2, \ldots, q_n\}$. Then we compute error of fitting, i.e., squared distance of each point between the original data and the approximated data, $d_i^2 = |p_i - q_i|^2, \quad 0 \le i \le n$. From the maximum d_i^2 we compute maximum squared distance, $\xi^{max} = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$. If maximum squared distance of any j^{th} segment is greater than ξ^{lmt} then this segment is splitted (replaced) with two new segments, j_1^{th} and j_2^{th}, i.e., $S = \left[\{S\} - \{S_j\}\right] \bigcup \{S_{j_1}, S_{j_2}\}$. A new breakpoint bp_{new} from original data is added in the set of breakpoints where the error is maximum, i.e., $BP = \{BP\} \bigcup \{bp_{new}\}$. For example, if segment S_1 splits at p_6 then a new breakpoint $bp_{new} = p_6$ is inserted between breakpoints p_1 and p_{13} ($BP = \left\{ p_1, \overbrace{p_6}^{bp_{new}}, p_{13}, p_{25}, p_{37}, \ldots, p_n \right\}$) and two new segments $\{p_1, p_2, \ldots, p_6\}$ and $\{p_6, p_7, \ldots, p_{13}\}$ replace S_1. The fitting process is repeated with new set of breakpoints until mean square error of each segment is equal to or less than ξ^{lmt}.

The above described fitting process is applied to color or luminance variations in temporal dimension of each spatial location separately. It is obvious that each spatial locations have different number of breakpoints (control points) selected from different frames, in other words rectangular shape of video frames is no longer intact.

Section 4 describes the proposed algorithm formally for a single spatial location. In order to have a better comparative look of fitting process between Catmull-Rom spline (CRS) and quadratic Bézier curve (QBC), we marked steps in curly braces where CRS and QBC differs in fitting. Figure 6 and Figure 7 show fitting of luminance values of in 80 frames of a video by Catmull-Rom spline and quadratic Bézier curve, respectively.

4 ALGORITHM

```
procedure Fitting algorithm for luminance variations of single spatial location.
spatial location $(x,y)$, $1 \le x \le W$, $1 \le y \le H$.
Luminance of $(x,y)$ in frame 1 to $n$ is $\{p_1, p_2, \ldots, p_n\}$

Require: :
    Max. allowed squared distance i.e., $\xi^{lmt}$,
    Breakpoints interval i.e., $\delta$,
    Points of original data: $O = \{p_1, p_2, \ldots, p_n\}$,
    Breakpoints: $BP = \{p_1, p_{1+\delta}, , p_{1+2\delta}, \ldots, p_n\}$,
1:  Get indices of BP: $V = \{1, 1+\delta, 1+2\delta, \ldots, n\} = \{v_1, v_2, \ldots, v_u\}$,
2:  Divide $O$ into segments $S = \{S_1, S_2, \ldots, S_{u-1}\}$, $S_i = \{p_{v_i}, \ldots, p_{v_{i+1}}\}$.
3:  { Not required                                                  CRS
      Find MCP of each segment                                      QBC
      $MCP = \{P_1^{S_1}, , P_1^{S_2}, \ldots, P_1^{S_{u-1}}\}$     QBC
4:  Fit a spline to each segment, i.e., Find $Q = \{q_1, q_2, \ldots, q_n\}$
5:  $d_i^2 = |p_i - q_i|^2, \quad 0 \le i \le n$
6:  $\xi^{max} = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$, $\xi^{max} \in k^{th}$ frame, $k \in j^{th}$ segment
7:  while $\xi^{max} > \xi^{lmt}$ do
8:      $bp_{new} = p_k$
9:      $V = \{V\} \bigcup \{k\}$
10:     Split $j^{th}$ segment into $j_1^{th}$ and $j_2^{th}$ segments
11:     $BP = \{BP\} \bigcup \{bp_{new}\}$
12:     $S = \left[\{S\} - \{S_j\}\right] \bigcup \{S_{j_1}, S_{j_2}\}$
13:     { Not required                                              CRS
          Update $MCP =$
          $\{MCP\} \bigcup \{P_1^{j_1}, P_1^{j_2}\}$                QBC
14:     Using updated BP fit spline/curve to:
        { $j-1$, $j_1$, $j_2$, $j+1$ segments    CRS
          $j_1$, $j_2$ segments                  QBC
15:     Find $d_i^2$ of:
        { $j-1$, $j_1$, $j_2$, $j+1$ segments    CRS
          $j_1$, $j_2$ segments                  QBC
```

```
16:     Update $\xi^{max}$=
          $\begin{cases} Max\left(\xi^{max}, d^2_{j-1}, d^2_{j_1}, d^2_{j_2}, d^2_{j+1}\right) & \text{CRS} \\ Max\left(\xi^{max}, d^2_{j_1}, d^2_{j_2}\right) & \text{QBC} \end{cases}$
          $\xi^{max} \in k^{th}$ frame, $k \in j^{th}$ segment
17:   Find count of interpolating points (C) from $V = \{v_1, v_2, \ldots, v_u\}$, $u \lll n$:
        $C = \{c_1, c_2, \ldots, c_{u-1}\}$, $c_i = v_{(i+1)} - v_i + 1$
18:   $\begin{cases} \text{Encode } BP \text{ and } C & \text{CRS} \\ \text{Encode } BP, MCP \text{ and } C & \text{QBC} \end{cases}$
```

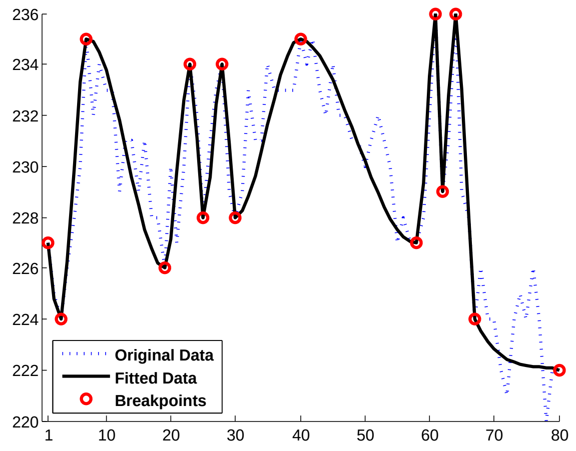

Figure 6: Catmull-Rom spline fitting to luminance values in 80 frames of a video, $\xi^{lmt} = 21$, $\delta = 79$, PSNR=43.845-dB.

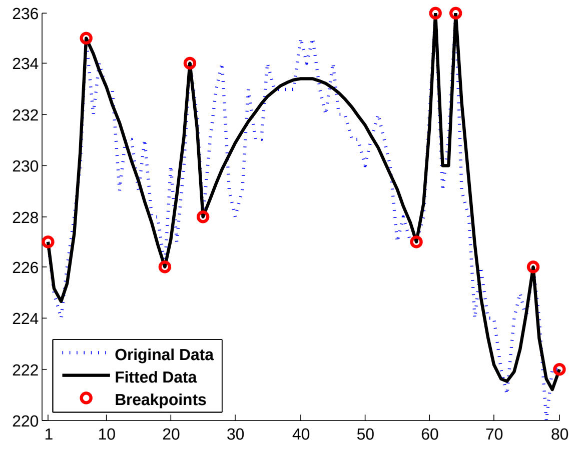

Figure 7: Quadratic Bézier curve fitting to luminance values in 80 frames of a video, $\xi^{lmt} = 21$, $\delta = 79$, PSNR=45.115-dB.

5 EXPERIMENTS AND RESULTS

We have applied the proposed algorithm on various naturally recorded and synthetically created video sequences. Output data is entropy encoded. The performance of proposed method is evaluated in terms of *bit-rate*, measured in bits per pixel (bpp) and PSNR, measured in decibel (dB). Table 1 gives the details of selected input video sequences whose results are presented in this paper. *Salesman* and *Foreman* are naturally recorded luminance videos at 8-bpp, while *Darius* is a synthetically created RGB video at 24-bpp. Table 2 compares the performance of CRS and QBC with Three Step Search method (TSS) [KIH$^+$81], a well known and widely used block matching algorithm for temporal compression of video data. For TSS every alternative frame is predicted from previous frame (reference frame), macro block size is 8×8, and range of search window is ± 8 in both horizontal and vertical directions. Reference frames are first differentially then entropy coded along with motion-vectors (MVs). Predicted frames are not coded because they are approximated from reference frames. The fitting methods achieved better compression performance than TSS, because the fitting work of our algorithm is at pixel level and they approximate the variations of luminance with good level of accuracy. We did not compare the performance for RGB video, because block matching algorithm convert the RGB data to YC_bC_r data, and block matching is performed for only luminance data (Y), while chrominance data (C_bC_r) is sub-sampled (4:2:0), and predicted chrominance frames are obtained from MVs of luminance data. Therefore comparison of CRS and QBC with TSS for RGB video is not appropriate.

Figures 8, 9 and 10 show R-D curves of CRS and QBC for *Salesman*, *Foreman* and *Darius* sequences respectively. Note that these statistics are only for temporal compression of video data; neither spatial compression nor quantization is used. The statistics show that the proposed method yields good objective performance for naturally recorded videos (*Salesman* and *Foreman*) and very good performance for synthetically created video (*Darius*). For the *Salesman* sequence, around 35-dB PSNR is achieved at bit-rate 0.59-bpp and 0.64-bpp by CRS and QBC respectively. For the *Foreman* sequence, around 35-dB PSNR is achieved at bit-rate 1.98-bpp and 2.1-bpp by CRS and QBC respectively. *Darius* is a color sequence and bit-rate of original *Darius* sequence is 24-bpp. We achieved around 35-dB PSNR at bit-rate as low as 1.5-bpp by both CRS and QBC.

Now look at the comparative performance of CRS and QBC. For the *Salesman* sequences at low bit-rate CRS performs better but at bit-rate>0.82-bpp QBC leads CRS. For the *Foreman* sequences CRS performs better than QBC at all values of bit-rate. For the *Darius* sequence the curve is quite interesting, at bit-rate<1.6-bpp CRS performs better, 1.6-bpp<bit-rate<3.5-bpp QBC performs better, and bit-rate<3.5-bpp, again CRS takes leads over QBC.

Figures 11, 12 and 13 show 30th frames of CRS and QBC encoded videos for *Salesman*, *Foreman* and *Darius* video sequences respectively. It can be seen from these figures that subjective performance i.e., *human vi-*

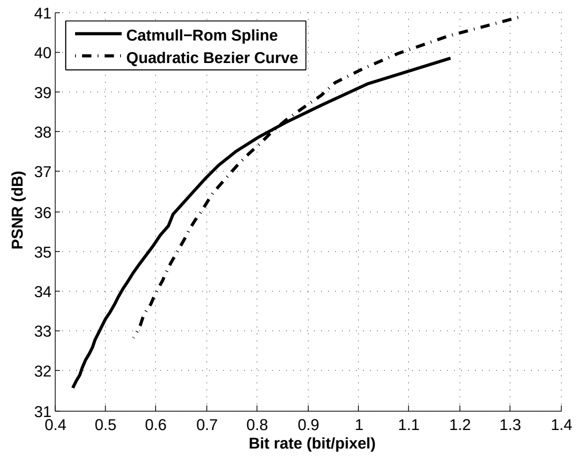

Figure 8: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Salesman* sequence.

sual acceptance of CRS and QBC approximated video frames is very good and no blocking artifacts are produced.

6 DISCUSSION

***Split of a segment*:** Lines 8 to 16 are important steps of our fitting algorithm. For the correct and efficient implementation of our algorithm it is very important to find which other segments are affected by splitting of a segment. Let us assume that an arbitrary segment j splits into two segments j_1 and j_2. For the quadratic Bézier curve we have to recompute *middle control points*, approximating points and squared distances of j_1 and j_2 segments only. But for Catmull-Rom spline the situation is not so simple. For three consecutive segments $j-1$, j and $j+1$, the last three control points of $(j-1)^{th}$ segment i.e., P_{j-1}, P_j and P_{j+1} are same to first three control point of j^{th} segment and first control point of $(j-1)^{th}$ segment i.e P_{j-2} is not shared with j^{th} segment. Similarly the first three control points of $(j+1)^{th}$ segment i.e., P_j, P_{j+1} and P_{j+2} are same to last three control points of j^{th} segment and last control point of $(j+1)^{th}$ segment i.e P_{j+3} is not shared with j^{th} segment. This means that a segment shares control points with its previous and next segments. Consequently adding a new breakpoint (splitting) at j^{th} segment requires to recompute approximating values and squared distances for two new segments, i.e., j_1 and j_2, obtained from splitting of j^{th} segment, in addition to that approximating values and squared distances of previous and next segments of j^{th} segments, i.e., $(j-1)^{th}$ and $(j+1)^{th}$ segments are also need to be recomputed. Fortunately Catmull-Rom spline saves some computation by not requiring least square solution as required by quadratic Bézier curve to find the value of its *middle control point*. Lines 3 and 13 of the fitting algorithm describe these recomputation steps for quadratic Bézier curve and Catmull-Rom spline.

***Rate Control*:** Rate of the output data is controlled by varying the value of ξ^{lmt}. Increasing the value of ξ^{lmt} decreases the bit-rate. The default value of ξ^{lmt} is 100.

Video Name	Format	Number of Frames	Bit-rate
Salesman (luminance)	CIF 352×288	45	8-bpp
Foreman (luminance)	SIF 352×288	44	8-bpp
Darius (RGB)	SIF 352×288	44	24-bpp

Table 1: Details of input video sequences.

Method Name	*Salesman*		*Foreman*	
	PSNR	Bit-rate	PSNR	Bit-rate
TSS	38.132	1.7768	35.875	2.7509
CRS	38.244	0.8574	35.589	2.0687
QBC	38.291	0.8578	35.812	2.2516

Table 2: Performance comparison of TSS [KIH+81], CRS and QBC.

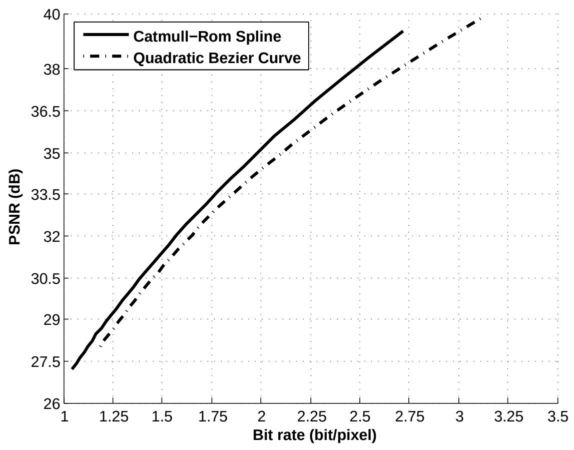

Figure 9: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Foreman* sequence.

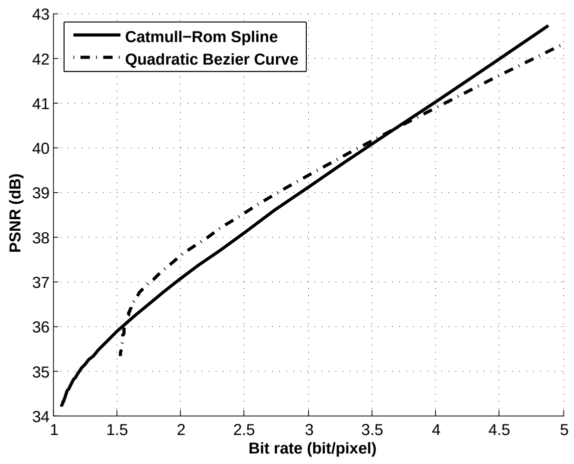

Figure 10: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Darius* sequence.

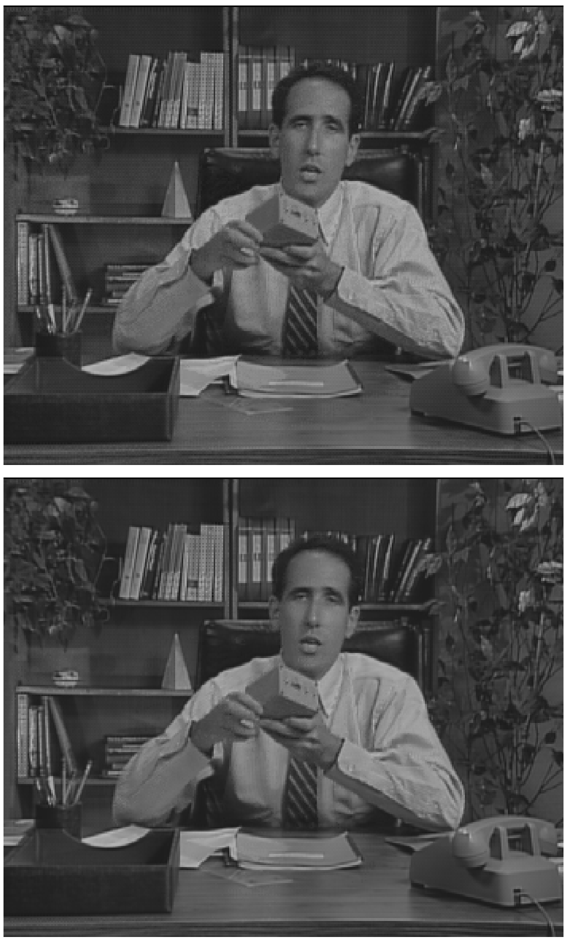

Figure 11: 30th frame of *Salesman* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 38.68-dB, 0.92991-bpp. Bottom: QBC approximated frame, 39.968-dB, 1.0796-bpp.

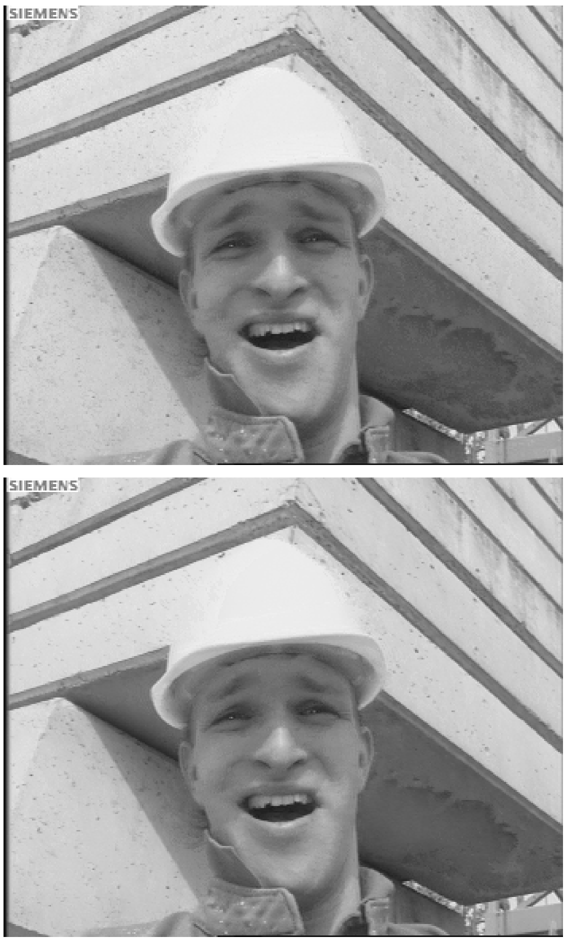

Figure 12: 30th frame of *Foreman* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 37.591-dB, 2.401-bpp. Bottom: QBC approximated frame, 38.191-dB, 2.7374-bpp.

***Output data encoding requirement*:** For CRS we need to encode (1) breakpoints (*BP*) and (2) count of interpolating points (*C*). For QBC, in addition to (1) and (2), we also need to encode *middle control points* (*MCP*). If video data is fitted using m segments then for CRS, $m+1$ values of *BP* and m values of *C* are need to be encoded. For QBC with equal number of segments, $m+1$ values of *BP*, $m+1$ values of *MCP* and m values of *C* are need to be encoded. Apparently CRS has less output data to be encoded. But it also depends on how much splitting of segments occurs. Usually splitting of a segment by CRS fitting is more often than splitting of a segment by QBC fitting.

***Reasons to choose quadratic Bézier curve and Catmull-Rom spline*:** (1) Both QBC and CRS are computationally efficient than other curves e.g., Natural cubic spline, B-spline, cubic Bézier curve etc. In fact we also used Natural cubic spline and cubic Bézier curve and found them less feasible than QBC and CRS. (2) Breaking of a segment into two segments keeps the computation cost within acceptable limit for both QBC and CRS.

Computational cost: QBC is computationally more efficient than CRS, because, (1) QBC is a quadratic function, while CRS is cubic a function. (2) Due to the least square fitting, QBC approximates longer segments, which means it causes lesser splitting and needs lesser computation than CRS.

***Naturally recorded vs synthetically created videos*:** The latter has less noise and more uniform distribution of luminance/color. Therefore, both CRS and QBC perform extremely well for synthetically created videos.

***Choosing between CRS and QBC*:** CRS and QBC behave quite similar, but if higher compression is desired then CRS is slightly better than QBC. If computational efficiency is of more importance then QBC is more appropriate choice.

***Limitations*:** The performance of fitting process degrades for spatial locations where changes in luminance in temporal dimension is very sharp. Because such spa-

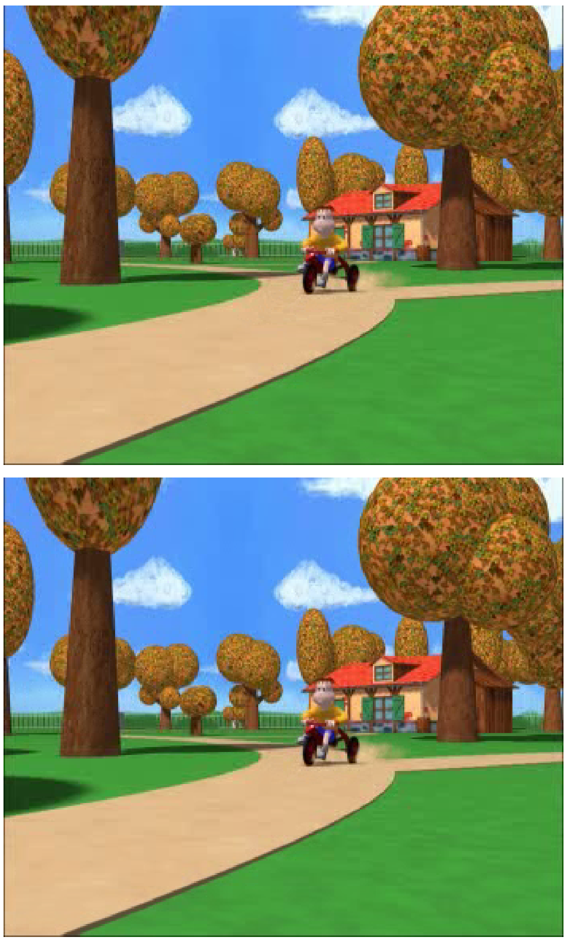

Figure 13: 30th frame of *Darius* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 41.838-dB, 4.4234-bpp. Bottom: QBC approximated frame, 42.272-dB, 4.9813-bpp.

tial locations cause lot of splitting of segments. Due to non-rectangle shape of output data, combining spatial compression with temporal compression needs special care and shape-adaptive wavelet transform can be used for spatial compression.

7 CONCLUSION

We presented a new method for compression of temporal video data by Catmull-Rom spline and quadratic Bézier curve fitting. Detail of fitting strategy and pseudo code of the algorithm are presented. We tested the proposed method using luminance and color (RGB) temporal data of naturally recorded and synthetically created video sequences. Experimental results show that the proposed method yields very good results both in terms of objective and subjective quality measurement parameters, i.e. bit-rate/PSNR and *human visual acceptance*, without causing any blocking artifacts. The method is suitable for compression of both naturally recorded and synthetically created videos such as animations and cartoons.

8 FUTURE WORK

Compression of spatial video data by spline/curve fitting is under investigation.

REFERENCES

[BBB95] Richard H. Bartels, John C. Beatty, , and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1995.

[CP02] Chun-Ho Cheung and Lai-Man Po. A novel cross-diamond search algorithm for fast block motion estimation. *IEEE Trans. Circuits And Systems For Video Technology*, 12(12):1168–1177, December 2002.

[CR74] Edwin Catmull and Raphael Rom. *A Class of Local Interpolating Splines, Computer Aided Geometric Design*. Academic Press, San Francisco, 1974.

[Gha03] Mohammed Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding*. Institution Electrical Engineers, new edition, 2003.

[IM00] Prakash Ishwar and Pierre Moulin. On spatial adaptation of motion-field smoothness in video coding. *IEEE Trans. Circuits And Systems For Video Technology*, 10:980–989, September 2000.

[IO93] Koichi Itoh and Yoshio Ohno. A curve fitting algorithm for character fonts. *Electronic Publishing*, 6(3):195–198, 1993.

[KB84] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18(3):33–41, 1984.

[KIH$^+$81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. *Proc. Nat. Telecommun. Conf., New Orleans, LA*, pages G5.3.1–G5.3.5, December 1981.

[LCT07] Tsung-Ching Lin, Shi-Huang Chen, and Trieu-Kien Truong. Medical image compression using cubic spline interpolation with bit-plane compensation. *Proceedings of the SPIE, Medical Imaging 2007: PACS and Imaging Informatics, San Diego, California, USA*, 6516:65160D, February 2007.

[LV07] Váša Libor and Skala Václav. Coddyac: Connectivity driven dynamic mesh compression. In *3DTV Conference Proceedings*, 2007.

[Say05] Khalid Sayood. *Introduction to Data Compression*. Morgan Kaufmann, third edition, 2005.

[SD02] Somphob Soongsathitanon and Satnam S. Dlay. A new orthogonal logarithmic search algorithm for fixed block-based motion estimation for video coding. *Proceedings of Third International Symposium on Communication Systems Networks and Digital Signal Processing, Sheffield Hallam Univ. Press Learning Centre*, pages 256–256, 2002.

[Thy05] KS Thyagarajan. *Digital Image Processing with Application to Digital Cinema*. Focal Press, 2005.

[UM00] Yoke Khim Ung and Farzin Mokhtarian. Multi-scale spline-based contour data compression and reconstruction through curvature scale space. *Proceedings ICASSP '00, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6:2123–2126 vol.4, June 2000.

[WOZ01] Yao Wang, Jôrn Ostermann, and Ya-Qin Zhang. *Video Processing and Communications*. Prentice Hall, first edition, 2001.