Automatic Recognition of Camera Zooms

Stephan Fischer, Ivica Rimac, and Ralf Steinmetz

1
Industrial Process and System Communications
Department of Electrical Engineering and Information Technology
Technical University of Darmstadt
Merckstr. 25 • D-64283 Darmstadt • Germany

Abstract. In this paper we explain how camera zooms in digital video can be recognized automatically. To achieve this goal we propose a new algorithm to detect zooms automatically and present experimental data on its performance.

1 Introduction

In this paper we describe a new algorithm to recognize zooms automatically. Besides only recognizing that a zoom has taken place it is often very useful to compute an exact scaling factor of a frame. We developed a new algorithm which both recognizes zooms and computes a scaling rate. The algorithm can be used to derive semantic features of digital film, such as automatic scene groupings or trailers [Fis97], [BR96]. The paper is structured as follows: Following a review of related work in section 2. In section 3 we propose a new algorithm to recognize camera zooms. In section 4 we present experimental results we obtained using our algorithm. Section 5 concludes the paper and gives an outlook.

2 Related Work

The recognition of zooms as well as the calculation of zoom factors has been investigated in the context of self-calibration of cameras in the past. [PKG98] describe a versatile approach which can be used to self-calibrate a camera. The authors specialize the general case towards scenarios where the focal length varies. The difference to the work described in this paper is that our algorithm is much simpler and hence faster to compute. The same statement is true for the work described by [HA97] and [FLM92] which. Tse and Bakler describe a global zoom/pan estimation algorithm in [TB91], which requires analysis of a high-density field of motion vectors. Since determining an entire frame of motion vectors with high spatial resolution is a very time consuming process, Zhang et al. [ZGST94] propose the block-matching algorithm for computing the motion vector field. But this method is inherently inaccurate when there is multiple

This work is supported in part by Volkswagen-Stiftung, Hannover, Germany.

motion inside a block. Furthermore, block-matching is particularly problematic during a zoom sequence because the area covered by the camera angle expands or contracts during a zoom. Finally, the motion of a large object in a camera shot proved to be a major source of error for camera pan detection.

3 Zoom Detection

A *zoom operation* is defined by an enlargement of the content of video frames towards the center of the zoom (zoom-in) or by a reduction (zoom-out). The overall size of the frames remains constant during this operation. A zoom-in is shown in Figure 1. The arrows specify that the image is the result of a zoom-in from an earlier image.

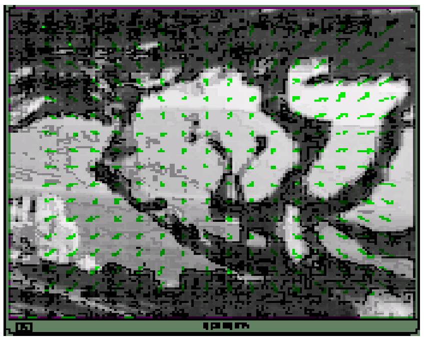


Fig. 1. Example of a zoom operation.

- The *zoom factor* is the factor by which an image has to be scaled to obtain the next image of a video.

3.1

3.2 Zoom Recognition based on Spline Interpolation

The approach we propose uses splines to recognize the fact that a zoom took place as well as to calculate a zoom factor. It is not our goal to calculate the self-calibration of a camera, however, we merely compute the zoom effect. The principal idea is that in any zoom-in or zoom-out sequence either an image *A* can be found as part of an image *B* or an image *B* can be found in an image *A*, if *A* is preceding *B* in time. The task should therefore be to use a mathematical approach to find an image in another. This can be achieved using the spline interpolation of images.
Spline interpolation serves to interpolate a set of discrete values by a curve or by a surface in order to model a continuous function. To interpolate images a variety of different approaches to spline interpolation are well known, for example cubic polynomial splines, non rational uniform B-splines (NURBS) or thin-plate splines.

In our approach we used natural cubic splines. Two important advantages of the spline approach can be identified: First, no computation of motion vectors is needed. Motion vectors themselves already contain a significant amount of errors caused by camera, digitalization and calculation artifacts also influencing the following zoom recognition process. Secondly a precise zoom factor indicating the amount of enlargement or reduction in size of an image can be computed.
The approach we propose is to transform two images *A* and *B* of a video sequence to a spline representation and then to stretch the spline representing image *A* in order to locate the spline representing image *B* and vice versa. Both directions have to be applied as it is unknown in advance if a zoom-in or a zoom-out is to be found and as it is not sure if an image is contained totally in another image. By an adaptation of a variable scaling factor the real zoom factor can be computed.
As the exact mathematical formulation of splines is beyond the scope of this paper we start to explain our method being applied for one-dimensional images thus simplifying the understanding of the underlying processes. We then extend the algorithm to two-dimensional images.

3.2.1 Zoom Recognition in One-dimensional Images

In the case of one-dimensional images two arrays *A* and *B* can be used containing values representing the brightness of a specific pixel of an image. Let *B* be an image which is the result of a zoom operation and thus some sort of transformation of image *A* (see Figure 2).

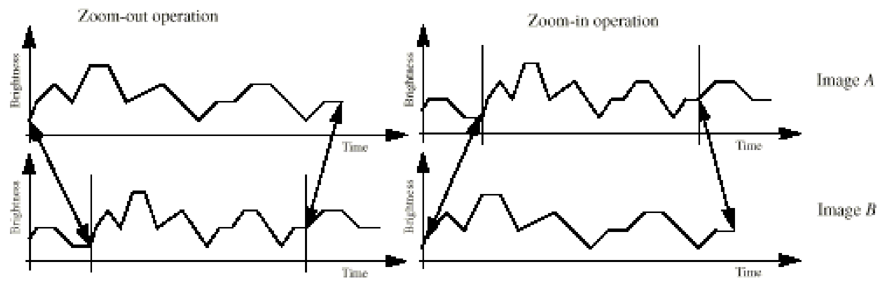


Fig.2. Zoom operations.

Representing images *A* and *B* as splines a spline *B* can be located in a spline *A* if image *B* is a video frame to be presented later in time than frame *A* (zoom-in operation). In the case of a zoom-out operation a search of spline *A* in spline *B* will be successful. A zoom-out can thus be recognized searching for image *A* in image *B*. A great advantage of this representation is that the search can be performed very efficiently as only a limited set of spline values have to be compared and as the comparison is not bound to values lying on the pixel grid. The algorithm to compute the zoom factor of a pair of images *A* and *B* is shown in Figure 3.

```
Calculate splines representing images A and B.
Search pattern of spline B in spline A.
```

```
  If pattern can be found then result := zoom-in.
  Else search pattern of spline A in spline B.
  If pattern can be found then result := zoom-out.
  Else result := no zoom.
```

Fig. 3. Zoom recognition in 1D-images.

In a first step the splines representing images *A* and *B* have to be calculated. A very efficient way to compare splines is to use the second derivative of the images which can be computed applying a Laplace-operation. The use of the second derivative guarantees that uniform areas of the images as well as regions of a constant gradient disappear speeding up the calculation of the splines as well as the matching process between the images. As the Laplace-operation calculates the second derivative the zero values correspond to the turning points in the images if and only if the first derivative is nonlinear. Searching for turning points to match *A* and *B* the following two conditions are possible: image *A* is contained in image *B* entirely in a smaller form, or image *A* is only partially contained in image *B* due to fast camera panning.
In the first case a potential zoom factor *lambda* can be estimated using the following formula:

$$\lambda = \frac{\overline{TP_{1,2}TP_{1,1}}}{\overline{TP_{2,2}TP_{2,1}}}$$

where the numerator denominates the turning points (TP) in image *A*. After estimating *lambda* the validity has to be checked by picking x-values randomly and by verifying

$$\text{spline}_A(x) = d + \text{spline}_B(\lambda x)$$

where d is the distance by which spline *A* has been translated in spline *B*. If the verification fails the next pair of turning points TP2,2 and TP2,1 is used to compare the images. This step is repeated until *A* has been located in *B*. It should be noted that the matching process is very fast as only those values have to be compared where both splines are equal. As a result a zoom factor greater than 1 denotes a zoom-out while a value less than 1 denotes a zoom-out.
In the second case a verification can fail if the selected turning point of image *A* is not contained in image *B* anymore due to fast camera panning. To solve this problem a new pair of turning points has to be chosen in spline *A* similar to the choice of new points in spline *B*. This has to be repeated until the spline has been located. In the case where no zoom has been recognized we assume that no zoom is contained in the video.

```
Calculate splines representing images A and B.
SplineMatch := FALSE, result := undefined
select first turning point (TP) in A
while ((SplineMatch == FALSE) AND (turning points in A
unvisited))
    select second turning point in A
    while ((SplineMatch == FALSE) AND (turning points in
      B unvisited))
        find turning point (TP) in B using zero-
           crossings (second derivative)
        calculate zoom factor using second TP and
          evaluate result by testing spline equality with
            x- and y-direction
        if zoom factor can be calculated
            SplineMatch := TRUE
            result := zoom
        else select new base TP in B
    select new base TP in A
if (result == undefined)
    result := no zoom.
```

Fig. 4. Zoom recognition in 2D-images.

3.2.2 Zoom Recognition in Two-dimensional Images

The spline comparison is much more complex in the two-dimensional case. The search for a spline equality based on rows or columns is hereby a problem as rows or columns of image *A* are not only shrunk but also displaced towards the zoom center in image B in case of a zoom-in. Also the target coordinates of the displacement don't have to be integers. To solve this problem we propose the following algorithm which is only applied to one direction (either x or y) while we use the complementary axis to validate the search results (see Figure 4).
The search for turning points by localizing the zeros of the Laplace image cannot be performed by checking the zeros at the whole-numbered values of the spline solely as zeros between those could be missed. To avoid this problem zero crossings have to be examined. In the case of a change of the sign between two values of the second derivative a zero must have occurred between these. Applying this technique also turning points can be localized which are not located on the pixel grid. The algorithm first tries to match the images using turning points on the horizontal axis. In case of a failure the y-direction is examined. To speed up the algorithm we start the localization process in image *B* with x- or y-values corresponding to those of the turning point in image *A* and extend the search radius as the algorithm continues its execution.

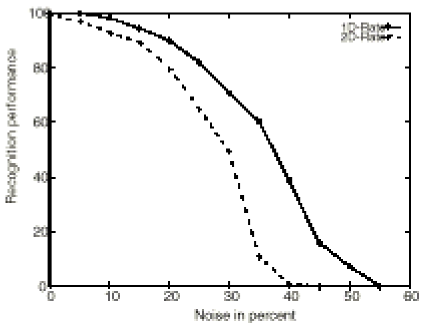
4 Experimental Results

We tested the spline algorithm with synthetical as well as with real image sequences. The performance has been defined as the amount of recognized zooms in relation to the length of the sequence. Our experimental data contained sequences without camera panning and zooms but with object movement (control instance), sequences containing

zooms solely, sequences with zoom and camera panning and sequences with zooms, camera panning and object movement.
We also tested the robustness of the algorithm by creating an artificial noise of variable strength in the source images. Video images very often include a specific amount of noise caused by camera and compression errors.
In a first set of experiments we examined the performance of the algorithm applied to one-dimensional synthetic image sequences. The results are listed in Table 1.

Tab. 1. 1D zoom detection

Experiment	Zoom recognized [%]	False positive [%]
Sequences without camera operations	0	0
Sequences with zoom only	100	0
Sequences with zoom and camera panning	100	0
Sequences with zoom, camera panning and strong object motion	84.29	11.93



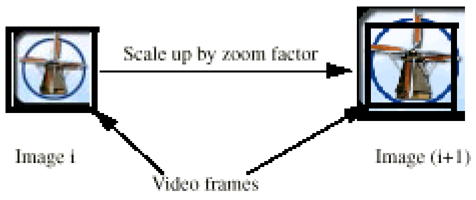


Fig. 5. 1D-zoom recognition in the presence of noise

Fig. 6. Generation of synthetical images

It is not surprising that a zoom cannot be recognized in the presence of strong object movement anymore. Experiments have shown that moving objects which are larger than 23 percent of the size of the total image cause a distortion which makes a zoom recognition impossible. This result was obtained generating synthetical image sequences containing moving objects and increasing their size step by step.
Surprisingly, the size of the zoom factor does not influence the recognition rate at all. We identified slow zooms as well as fast zooms with an equal success. Figure 5 shows the impact of noise with respect to the experimental results. These experiments were conducted using sequences containing zooms and camera panning. It is not surprising that a significant amount of noise lowers the performance of the algorithm. A similar behavior can be observed applying the algorithm to two-dimensional images. The

results are slightly worse and can be explained with the efficiency of the localization which only approximates the exact location of the spline. The results for the zoom detection using 2D-images are shown in Table 2.

Tab. 2. Results of 2D-zoom recognition

Experiment	Zoom recognized [%]	False positive [%]
Sequences without camera operations with object movement	2.19	1.1
Sequences with zoom only	98.03	0
Sequences with zoom and camera panning	94.77	3.11
Sequences with zoom, camera panning and strong object motion	82.97	15.33

We tested the performance of the spline algorithm with 50 videos consisting of 1000 frames each. Concerning the synthetical videos we adjusted the value of the zoom factor according to the function resulting in zoom factors traversing the interval [0; 2a] periodically. Adjusting *a* and *t* we were able to create different zoom variations within the synthetical sequence. For a zoom-in we constructed the synthetical images in such a way that we scaled up an image *i* by the zoom factor and cut out the edge regions in order to obtain the new image *(i+1)* equal in size to image *i* (see Figure 6).
As all of our experimental sequences started with a zoom-in we buffered the images of the zoom-in and played them in reverse order to obtain the respective zoom-out. This way we were able to produce sequences consisting of periodic zoom-ins and zoom-outs. The zoom factor can be adjusted varying the argument of the sine function thus creating slower or faster zooms. The scaling of the synthetical zooms can furthermore be adjusted using the parameter *a*. The mean experimental values we measured with the synthetical clips and with the real-world clips are shown in Table 2. A result of the experiments is that errors occurring during the zoom recognition process are mainly due to noise included in the images. This can in particular be shown applying some sort of smoothing, for example a Gaussian filter before calculating the splines. The efficiency can then be further improved. It turns out that a zoom recognition using splines outperforms the analysis of motion vector fields (both optical flow and MPEG vectors). Errors caused by noise in the original images and by object motion lower the performance of the algorithm. Considering object motion however it should be possible to find regions in an image where the distortion caused by the moving object is rather small. These areas can then be used to increase the performance of the algorithm.

5 Conclusions and Outlook

In this paper we propose a new method to recognize zooms automatically which also calculates a precise scaling factor between successive frames of a video sequence.

In our experimental section we have shown that the algorithm is has a reliability greater than 80 percent. Although the localization algorithm is quite robust we currently think about an integration of our algorithm with algorithms to compute the camera calibration. Another problem any algorithm to recognize zooms faces is object movement. In another set of experiments we currently examine if the spline localization can be performed on selected parts of an image where no object movement takes place.

References

[BBBB93] R.H. Bartels, J.C. Beatty, K.S. Booth, E.G. Bosch, and P. Jolicoeur. *Experimental comparison of splines using the shape-matching paradigm.* ACM Trans. Graph. 12(3), pp. 179-208, 1993.

[BR96] J. S. Boreczky and L. A. Rowe. *A comparison of video shot boundary detection techniques*. Journal of Electronic Imaging, 5(2):pp. 122-128, 1996.

[BX94] C. Bajat and G. Xu. *NURBS approximation of surface / surface intersection curves*. Adv. Comput. Math.2(1), pp. 1-21, 1994.

[BFB84] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. *Performance of Optical Flow Techniques*. International Journal of Computer Vision, 12(1), 1984.

[Fi97] S. Fischer. *Feature combination for content-based analysis of digital film.* PhD thesis, University of Mannheim, 1997.

[FLM92] O. Faugeras, Q.-T. Luong, and S. Maybank: *Camera self-calibration: Theory and experiments*. Proc. ECCV, 1992.

[HA97] A. Heyden, K. Aström: *Euclidean Reconstruction from Image Sequences with Varying and Unknown Focal Length and Principal Point*, Proc. CVPR, 1997.

[Hoe89] M. Hötter. *Differential Estimation of the Global Motion Parameters Zoom and Pan*. Signal Processing, 16, pp. 249-265, 1989.

[ISM93] K. Illgner, C. Stiller, and F. Müller. *A Robust Zoom and Pan Estimation Technique*. Proceedings of the International Picture Coding Symposium PCS’93, Lausanne, Switzerland, 1993.

[LK81] B. Lucas and T. Kanade. *An iterative image registration technique with an application to stereo vision*. DARPA IU Workshop, pp. 121-130, 1981.

[MKG98] M. Pollefeys, R. Koch, and L. Van Gool: *Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters,* Proc. Of ICCV, 1998.

[TB91] Y.T. Tse and R.L. Bakler. *Global zoom pan estimation and compensation for video compression*. Proceedings ICASSP, pp. 2725-2728, 1991.

[ZGST94] H. Zhang, Y. Gong, S.W. Smoliar, and S.Y. Tan. *Automatic Parsing of News Video*. Proceedings of IEEE Conf. on Multimedia Computing and Systems, 1994.