Contour-based Classification of Video Objects

Stephan Richter, Gerald Kühne, Oliver Schuster

University of Mannheim, Germany
{richter,kuehne,schuster}@informatik.uni-mannheim.de

ABSTRACT

The recognition of objects that appear in a video sequence is an essential aspect of any video content analysis system. We present an approach which classifies a segmented video object based on its appearance (object views) in successive video frames. The classification is performed by matching curvature features of the contours of these object views to a database containing preprocessed views of prototypical objects using a modified curvature scale space technique. By integrating the results of a number of successive frames and by using the modified curvature scale space technique as an efficient representation of object contours, our approach enables the robust, tolerant and rapid object classification of video objects.

Keywords: video content analysis, video object recognition, shape analysis, curvature scale space

1 INTRODUCTION

The use of video as an information medium has become commonplace. To provide access to the information contained in video data, appropriate content analysis and indexing methods are necessary. The distant goal of research in automatic content analysis of continuous media is to enable functionality like that already existing for textual information retrieval. Various methods covering different aspects such as shot boundary detection, scene determination, text extraction, and human face detection have been developed in this field.[5, 8, 18, 20, 23]

The recognition of objects that appear in a video sequence constitutes another essential part of any video content analysis system. In general, object recognition can be addressed at different levels of abstraction. For instance, an object might be classifiable as a “cat” (object class), as a “Siamese cat” (subordinate level) or as “my neighbour’s cat” (individual object).

We present a contour-based approach to classifying a wide range of objects in video sequences on the object-class level. Curvature features of the contour of each two-dimensional appearance of an object in a video frame are calculated. These features are matched to those of views of prototypical video objects stored in a database. The final classification of the object is achieved by integrating the matching results for a number of successive frames. This adds reliability to our approach, since unrecognizable single object views occurring in the video sequence are insignificant with respect to the whole sequence.

The calculation of the contour description relies on the curvature scale space (CSS) method developed by Mokhtarian[11–13] for shape-based image retrieval. We extended this technique for the processing of video sequences and enhanced it by extracting additional information from the CSS image and developing a new matching algorithm.

Our approach is restricted by the assumption that a reliable segmentation of the object to be classified is available. Though, the segmentation of common objects in arbitrary scenes is still beyond the capabilities of an artificial system, there exist a number of algorithms that succeed under constrained conditions. [7, 16]

The remainder of the paper is organized as follows: Section 2 summarizes related work. Section 3 describes our recognition approach in detail. Experimental results appear in Section 4. Finally, Section 5 offers concluding remarks.

Figure 1. Extract from the two-dimensional views representing the object class "car".

2 RELATED WORK

Contour-analysis techniques have existed in computer science for some time now. One of the first overviews of algorithms in the area of shape analysis was published by Pavlidis as early as 1978.[17] He restricted his review to the analysis of "silhouettes", as he called shapes and contours of two-dimensional objects. Already in 1978, Pavlidis mentioned that shape analysis is "an enormous subject". More than twenty years later, the subject is still enormous, many new approaches have been tried, and some progress has been made. Pavlidis mentioned that it seemed to be possible to develop rigorous mathematical algorithms to analyse shapes and provide results similar to human perception. Our research found that no straightforward mathematical metric can be found which models human perception with regard to shape analysis.

A more recent general survey of shape-analysis techniques was done by Loncaric.[9] The survey contains a section which covers aspects of the human visual system. Some of the theories of visual forms from the field of psychology are introduced. Among them is the Gestalt theory, which assumes that form is perceived as a whole. Opposed to the Gestalt theory, several decomposition theories are mentioned. Nevertheless, the human visual system is not understood well enough to discard either the Gestalt theory or the decomposition theories. In the later sections of his paper, Loncaric introduces several shape-analysis techniques grouped into boundary scalar transform, boundary space domain, global scalar transform, and global space domain techniques. The method which we use is a boundary scalar transform technique.

A paper which compares human perceptual judgments with the results of seven different shape-matching algorithms was written by Scassellati, Alexopoulos and Flickner.[19] They asked 40 volunteers to match 20 query shapes to a database of about 1,400 images taken from the QBIC[3,14,15] project. The algorithm which was closest to the results obtained from the volunteers was the turning angle approach. The turning angle approach basically describes which direction needs to be used to travel the perimeter of a contour. Nevertheless, this approach was only best in seven out of 20 queries. These results support our statement that no straightforward mathematical metric can be found to model human shape perception.

One of the most recent works to present an overview and the state of the art in theory and practice of shape analysis is a book by Costa and Cesar.[6] This book provides a good introduction to the subject ranging from the basic mathematical concepts to the acquisition and preprocessing of shapes. Several concepts of shape representation and characterization are presented.

One of the more promising contour analysis techniques is the CSS method introduced by Mokhtarian.[1,2,11–13] The advantages of his method are that it is size and rotation invariant and robust to noise. In the next section, we briefly describe the CSS method and outline the architecture of our video object recognition system.

3 VIDEO OBJECT CLASSIFICATION

Our system for object classification consists of two major parts: (1) a database containing contour-based representations of prototypical video objects and (2) an algorithm for matching objects extracted from a video sequence with the database. The object representation and the database are discussed in Section 3.1, the matching algorithm in Section 3.2.

3.1 Object Representation

In general, there are two ways to generate object-related information. First, one could extract features from a 3D object model, and second, it is feasible to use two-dimensional views of an object, taken from different perspectives as basis. In cognitive psychology a number of theories have been developed with regard to

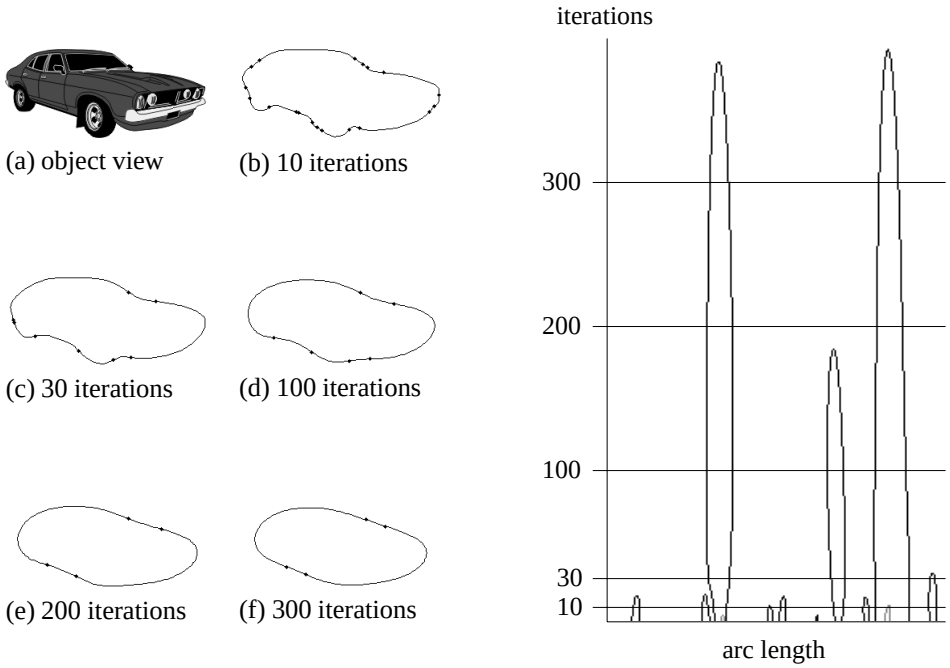

Figure 2. Construction of the CSS image. Left: (a)-(f) Smoothed contour after 10, 30, 100, 200 and 300 iterations. The small dots on the contour mark the curvature zero crossings. Right: Resulting CSS image.

object representation in the human brain.[4,10,22] Although a general theory is not available, psychophysical evidence indicates that humans encode three-dimensional objects as multiple viewpoint-specific representations that are largely two-dimensional.[21]

We adhere to this theory and store for each object class a number of different two-dimensional views, so-called object views. Furthermore, we pool different views of different objects into one object class to obtain a reliable class definition. Figure 1 illustrates the object class *car* by depicting several object views from this class. Of the views that can be generated from different perspectives, we prefer so called canonical views. A canonical view shows an object in a — with respect to human perception — typical perspective and provides a sufficient number of object characteristics to allow for rapid recognition. For instance, for a car, one possible canonical view is a slightly elevated view of the frontal and side parts of the object (see Figure 1). A view of the bottom of a car is generally not considered canonical.

Different sources of information are available to characterize a two-dimensional view (e. g. contour, colour, texture, motion, or relative location of the object to other objects). However, most common objects can be identified by their contours only.[22] In our approach, for each object view a few parameters are extracted from its contour and stored in conjunction with the object view's class name in a database. The parameters are calculated using a modified curvature scale space (CSS) technique.

3.1.1 Basic Curvature Scale Space Representation

The CSS technique[1,2,11–13] is based on the idea of curve evolution, i. e. basically the deformation of a curve over time. The technique provides a multi-scale representation of the curvature zero crossings of a closed planar contour. A zero crossing occurs, for instance, at the transition from a convex to a concave contour segment.

The contour is scanned iteratively for inflection points of the curvature while being smoothed by a Gaussian kernel. During the deformation process, zero crossings merge as transitions between contour segments of different curvature are equalized (see Figure 2). Consequently, after a certain number of iterations inflection points cease to exist and the shape of the closed curve is convex. Note that due to

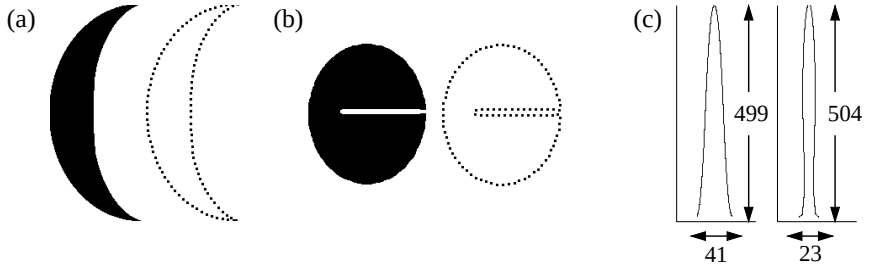

Figure 3. Ambiguities in CSS images. (a) shallow concavity: object view (left), contour (right) (b) deep concavity: object view (left), contour (right), (c) left: CSS image of (a), right: CSS image of (b).

the dependence on curvature zero crossings, convex object views cannot be represented with the CSS technique.

From the positions of the zero crossings at different scales, a so-called CSS image is constructed. The CSS image shows the zero crossings with respect to their position on the contour and the width of the Gaussian kernel (or the number of iterations, see Figure 2). Therefore, significant contour properties that are visible for a large number of iterations result in high peaks in the CSS image. However, areas with rapidly changing curvatures caused by noise produce only small local maxima.

To include size invariance into the CSS technique, we sample for each contour a fixed number of equidistant contour points (in our implementation we use 200 sample points).

In many cases the peaks of the CSS image provide a robust and compact representation of an object view's contour.[2,12,13] Note that a rotation of an object view on the image plane can be accomplished by shifting the CSS image left or right in a horizontal direction. Furthermore, a representation of a mirrored object view is obtained by mirroring the CSS image.

It is sufficient to extract the significant maxima (above a certain noise level) from the CSS image, i. e. selected for each maximum are its position on the contour and the value (iteration or Gaussian kernel width). For instance, for the example depicted in Figure 2, only four data pairs have to be stored, assuming a noise level of 30 iterations.

3.1.2 Modified Curvature Scale Space Representation

A main drawback to the basic CSS technique described in the last section is the occurrence of ambiguities. Under certain conditions shallow and deep concavities on a contour may result in peaks of the same height in the CSS image. Figure 3 depicts this problem: The shallow concavity of the object contour shown in (a) and the deep concavity of the object contour displayed in (b) result in peaks of nearly the same height (relative difference about 1 %) in the CSS images (c). Consequently, certain contours differing significantly in their visual appearance are claimed by the basic CSS technique to be similar.

Abbasi[2] presented several approaches to avoiding these ambiguities. However, the proposed strategies raise the computational costs significantly.

In our extension we utilize additional information already available in the CSS image. In addition to the height of a peak in the CSS image, we also extract the width at the bottom of the arc-shaped contour corresponding to the peak. As it is shown in Figure 3, the widths of shallow and deep concavities differ significantly creating CSS maxima of the same height (relative width difference is about 80 %). The width specifies the normalized arc length distance of the two curvature zero crossings enframing the contour segment represented by the peak in the CSS image.

Let us summarise our approach to mapping prototypical video objects to database entries. Each prototypical object is represented by a collection of object views. For the object views, in turn, a number of data triples consisting of positions, heights, and widths of the CSS maxima are stored in the database. The matching algorithm described in the following section utilises this information to compare extracted video objects and prototypical video objects.

3.2 Object Matching

Object matching is done in two steps. In the first, each individual object in a sequence is compared to all objects in the database by comparing peaks characterised by the triplets in the database. A list of the best matches is build for further processing. This first step is described in section 3.2.1. In the second step, the results from the first step are accumulated and a confidence value is calculated. Based on the confidence value, the object class of the object in the sequence is printed. The second step is described in section 3.2.2.

3.2.1 Object based Matching

In order to find the most similar object in the database compared to a query object from a sequence, a matching algorithm is needed. This algorithm is shown as Algorithm 1. The general idea of the algorithm is to compare the peaks in the CSS images of the two query objects cm1 and cm2 to each other based on the characterisation by the triplets of height, position on the arc and width. This is done by first determining the best position to compare the two images. It might be necessary to rotate or mirror one of the images so that the peaks are aligned best. Next, a matching peak is determined for each peak in cm1. If a matching peak is found, the Euclidean distance of the height and position of the peaks is calculated and added to the difference between the images. If no matching peak was found, the height of the peak in cm1 multiplied by a penalty factor is added to the total difference.

Several hygiene factors, which need to be met, exist. For all peaks, the matched peak needs to be within a certain position and width range. Only for the highest peaks, the height also needs to be within a certain range. The ranges are set via threshold parameters.

The matching algorithm has to take into account that the object from the image might be mirrored or rotated compared to the best match in the object-view database. Therefore, the matching algorithm needs to be executed multiple times until the best-matching position is found. A heuristic is used to shorten execution time. Only the most promising rotations are calculated. These are determined by shifting the CSS images so that the highest peaks of the CSS image are aligned. As mentioned before, shifting the CSS image corresponds to rotation of the original object. Since not all possible rotations of an object view are stored in the database, it is reasonable to compensate this shortcoming during the matching process. The algorithm which determines relevant shifting offsets is shown as Algorithm 2.

Algorithm 1 needs to be called several times to compensate for mirroring of the object in the sequence or the object view in the database. The call which results in the lowest difference is used for further processing

Algorithm 1 might return ∞, if e.g. the shift list is empty and therefore no adequate rotation could be found or if the highest maxima in the CSS images do not match within a given tolerance range. If this is the case, the two objects are significantly different and therefore a match is not possible. A clear rejection helps to improve the overall results of the matching algorithm, since object-views which do not bear much resemblance to objects from a sequence are eliminated for further evaluation in this way.

3.2.2 Sequence based Matching

Once the matching algorithm has been executed, a list of matches for each object in the sequence exists. This list contains the difference to the object view in the database and the object class of the object view. Only the top match, i.e. the object view with the least difference, is used for evaluation. It might be that the difference to the top match is ∞. If so, no reasonable match could be found in the database. Since the database does not contain all possible object views, such a result might occur frequently, depending on the object in the sequence. ∞ as difference therefore clearly indicates that no conclusive statement can be made about the class of the object from the sequence.

All top matches which were recognised are used for accumulation. In the accumulation process, the inverse difference for each object in the sequence is added to an entry for the specific recognised object class. This procedure yields a list which contains one value for each object class. The total of these floats gives the total accumulated difference for the processed sequence. Each entry in the list is divided by this total, resulting in a percentage number. The object class with a percentage higher than 70 % is considered to be the object class of the sequence. Higher percentage numbers stand for better recognition rates. Examples of test runs can be found in Section 4.

Algorithm 1 Matching of two CSS images

type $cssmaximum := \langle pos, height, width \rangle$
funct $matchCssImages(list\langle cssmaximum\rangle cm1, list\langle cssmaximum\rangle cm2) \equiv$
begin
var $mindifference := \infty$ return parameter
var $difference := 0$
sort cm1, cm2 according to height descending
$shiftlist = calculateShiftList(cm1, cm2)$ see Algorithm 2
foreach $offset \in shiftlist$ **do** try all reasonable rotations
shift cm2 by offset
while $unmatched\ maxima \in cm1$ **do** search a match for each maximum
take next maximum from cm1
$match\ found := false$
foreach $unvisited\ maxima \in cm2$ **do**
if $width_{cm1} + T_{width}\% > width_{cm2} > width_{cm1} - T_{width}\%$
and $pos_{cm1} + T_{pos}\% > pos_{cm2} > pos_{cm1} - T_{pos}\%$
then $match\ found := true$
$break$
fi
od
if $match\ found$
then if $(height_{cm1} > maxheight_{cm1} - 50\%$ if handling significant maximum
and $height_{cm1} + T_{height}\% > height_{cm2} > height_{cm1} - T_{height}\%)$
or $height_{cm1} \leq maxheight_{cm1} - 50\%$
then $posdiff := ||pos_{cm1} - pos_{cm2}||$
$heightdiff := ||height_{cm1} - height_{cm2}||$
$difference := difference + \sqrt{posdiff^2 + heightdiff^2}$
mark maxima in cm2 as visited
else $difference := \infty$ reject this shift offset
fi
else if $height_{cm1} > maxheight_{cm1} - 50\%$ no match, check for significant maximum
then $difference := \infty$ reject this shift offset
else $difference := difference + height_{cm1} * T_{penalty}$ add height and penalty
fi
od
foreach $unmatched\ maxima \in cm2$ **do**
$difference := difference + height_{cm2} * T_{penalty}$ add height and penalty
od
if $mindifference > difference$ check if better match is found
then $mindifference := difference$
fi
od
$return\ (mindifference)$
end.

Algorithm 2 Determining the Shift List

```
funct calculateShiftList(list⟨cssmaximum⟩cm1, list⟨cssmaximum⟩cm2) ≡
  var max := maximum height in cm1
  foreach height_cm1 ≥ max − 20% do
    foreach maximum ∈ cm2 do
      if height difference > 20% then continue fi      if height is not in tolerance, ignore
      if width difference > 20% then continue fi       if width is not in tolerance, ignore
      add position difference to list
    od
  od
  return(list⟨position differences⟩)
end.
```

4 EXPERIMENTAL RESULTS

Our test database contains five object classes containing *animals*, *birds*, *cars*, *people*, and *miscellaneous objects*. For each object class we collected 25 – 102 images from a clip art library. The clip arts are typical representatives of their object class with easily recognizable perspectives (canonical views). See Figure 1 for examples of prototypes of the object class *car*.

The object class *people* contains the most objects (102 images). The contours of humans differ greatly in image sequences, e. g. the position of the arms and legs makes a great impact on the contour. To make recognition of a query object possible, it is necessary to have a great variety of human images in the database. Otherwise Algorithm 1 will reject all images in the database and no results are obtainable. The object class *car* is very well represented in the database, too. With a limited number of 48 cars most perspectives and types are represented. The object classes *birds*, *animals*, and *miscellaneous objects* hold 25, 42, and 30 images.

The creation of the database with about 250 objects requires 30 seconds computation time on a standard personal computer*, thus 8 CSS images per second can be calculated. The database stores for each CSS image the name of the object class and the data of the relevant peaks (height, position, width).

Several short real-world video sequences were tested. Five sequences contain rigid objects (cars) and 6 sequences show non-rigid objects (people and one bird sequence). The results of these sequences are shown in table 1. For each sequence the type of segmentation (automatically or manually) and the number of frames are given. The length of each sequence ranges from 2 to 8 seconds. When an object enters or leaves a scene only parts of it are visible and its contour is heavily deformed. If this is the case the threshold parameter will reject all objects in the database: no match is possible. The column *matched frames* in table 1 shows the number of frames where the matching to at least one object in the database was successful.

To match a sequence to the database first the CSS features of the sequence need to be calculated. In a second step the calculated CSS features of the sequence are matched to the precalculated CSS features of the object views stored in the database. The whole matching process can be done in 5 frames per second.

The first 3 *people*–sequences are segmented automatically. Kim describes a segmentation method which calculates differences between a background image and the unknown frame based on edges. [7] The sequence *People–3* was automatically segmented using a level set based method described by Paragios. [16] The other sequences were segmented manually.

The sequence entitled *People–1* is a talking-head scene with small changes between the different frames. In *People–2* a human walks around and changes its orientation towards the camera. Only the upper part of the person is visible (see Figure 5 for sample images and best matches). In sequence *People–3* a human runs from a great distance to the camera.

* AMD Athlon 700 MHz CPU, 512 MB RAM

Sequence	Segmentation	Number of frames	Number of frames matched	Object class detected	
People–1	automatically	26	21	People	96 %
People–2	automatically	39	38	People	96 %
People–3	automatically	39	23	People	44 %
People–4	manually	29	26	People	71 %
People–5	manually	13	6	People	67 %
Bird–1	manually	15	10	People	57 %
Car–1	manually	51	33	Cars	100 %
Car–2	manually	21	11	Cars	87 %
Car–3	manually	51	40	Cars	100 %
Car–4	manually	19	14	Cars	100 %
Car–5	manually	22	17	Cars	100 %

Table 1. Results of the real-world sequences matched to the objects in the database. The variable parameters of Algorithm 1 are defined as follows: T_{pos}=20% , T_{height}=20% and T_{width}=40%

In all human sequences the object class *people* is the one with the most matches. The number of top matches ranges from 44 % to 96 %. Many contours in the sequence *People–3* have no similar representation in the database, so the relatively poor result of 44 % can be explained.

In the sequences entitled *Bird–1*, a dove walks from left to right through the image. In this sequence the correct object class cannot be detected. About one half of the 25 birds in the database fly with both wings visible. The contours of the other birds vary a lot (e. g. contours of a falcon, a hen, or a sparrow) and only few of them can be matched to the dove walking.

The third group of video sequences shows different cars crossing the image or moving around a corner (see Figure 4 for images of scene *Car–4* with best matches). The matching of cars is very successful (87 %–100 %) and the number of matched frames is relatively high. All cars enter or leave the scene, so images in the first or last frames exist, in which parts of the car are missing. These frames were rejected, which explains the few unmatched frames.

Sample images of the sequences *Car–4*, *People–2* and *People–5* with top matches are shown in Figures 4, 5, and 6. The parameters of Algorithm 1 are specified as follows: T_{pos}=20%, T_{height}=20% and T_{width}=40%. For the example depicted in Figure 6 one bad match is visible. The database contains no human in a similar pose and the CSS peaks of the helicopter and the human are just in range of the parameters. The fact that in this case the closest match to the human is a helicopter shows that the CSS method still has some shortcomings.

In addition to the good recognition of the object class, it is often possible to make out the perspective of the object in the frame. Comparing the top matches in Figures 4, 5, and 6 the perspectives of the objects in the sequence and database are often similar.

5 CONCLUSION AND OUTLOOK

The recognition of objects appearing in video sequences is one of the most challenging tasks in the field of automatic video content analysis. We presented a system of object classification that relies on a database containing preprocessed two-dimensional views of prototypical video objects. Classification is performed by matching curvature features of the presegmented video object in question to object view representations in the database.

In general, our system provides very promising results. The matching procedure operates fast and is mostly able to generalize individual objects to the database prototypes. Furthermore, the view-based approach enables the determination of the perspective in which the video object is captured.

While rigid objects (e.g. cars) are classified reliably, the performance of the system for non-rigid objects (e.g. human beings, bird) is inferior. A number of approaches are reasonable to improve the performance

Figure 4. Results for the sequence *Car–4*. Top: From left to right – segmented objects views from frames 7, 11, 15, 17 of the video sequence *Car–4*. Bottom: From left to right – best matches from the database for the object view displayed above.

Figure 5. Results for the sequence *People–2*. Top: From left to right – segmented objects views from frames 22, 26, 29, 32 of the video sequence *People–2*. Bottom: From left to right – best matches from the database for the object view displayed above.

on non-rigid objects. First of all, it is straightforward to extend the database and provide more prototypes covering different motions and postures. Second, the rotation invariance of the matching procedure could be restricted. Following our approach of using canonical views, it is reasonable to allow only a certain degree of rotation when matching object views to those in the database.

Finally, the CSS technique leaves room for improvement. In most cases peaks in the CSS image result from concave contour segments in the object view. However, under certain conditions convex segments may also result in CSS maxima. In the current implementation the CSS maxima extracted from the CSS image are not classified appropriately.

Future work may comprise the considerations mentioned above and the integration of reliable object segmentation techniques.

REFERENCES

1. Sadegh Abbasi and Farzin Mokhtarian. Shape similarity retrieval under affine transform: Application to multi-view object representation and recognition. In *Proc. International Conference on Computer Vision*, pages 450–455. IEEE, 1999.
2. Sadegh Abbasi, Farzin Mokhtarian, and Josef Kittler. Enhancing css-based shape retrieval for objects with shallow concavities. *Image and Vision Computing*, 18(3):199–211, 2000.
3. Jonathan Ashley, Ron Barber, Myron Flickner, James Hafner, Denis Lee, Wayne Niblack, and Dragutin Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in qbic. In *Proceedings of the SPIE*, volume 2420, pages 24–35, 1995.

Figure 6. Results for the sequence *People–5*. Top: From left to right – segmented objects views from frames 200, 204, 209, 225, 228 of the video sequence *People–5*. Bottom: From left to right – best matches from the database for the object view displayed above.

4. Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94:115–147, 1987.
5. Shih-Fu Chang, William Chen, and Hari Sundaram. Videoq: A fully automated video retrieval system using motion sketches. In *Proceedings Fourth IEEE Workshop on Applications of Computer Vision*, 1998.
6. Luciano da Fontoura Costa and Roberto Marcondes Cesar, Jr. *Shape Analysis and Classification*. CRC Press, 2000 NW Corporate Blvd, Boca Raton, FL 33431-9868, September 2000.
7. Changick Kim and Jenq-Neng Hwang. An integrated scheme for object-based video abstraction. In *Proceedings ACM Multimedia*, 2000.
8. Huiping Li and David Doermann. Text enhancement in digital video using multiple frame integration. In *Proceedings ACM Multimedia*, pages 19–22, 1999.
9. Sven Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, August 1998.
10. David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, CA, 1982.
11. Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):539–544, 1995.
12. Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Proc. International Workshop on Image DataBases and MultiMedia Search*, pages 35–42, 1996.
13. Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *British Machine Vision Conference*, 1996.
14. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Tabuin. The qbic project: Querying images by content using color, texture, and shape. In *Proceedings of the SPIE*, volume 1908, pages 173–187, 1993.
15. Wayne Niblack, Xiaoming Zhu, James Lee Hafner, Tom Breuel, Dulce Ponceleon, Dragutin Petkovic, Myron Flickner, Eli Upfal, Siegfredo I. Nin, Sanghoon Sull, Byron Dom, Boon-Lock Yeo, Savitha Srinivasan, Dan Zivkovic, and Mike Penner. Updates to the qbic system. In *Proceedings of the SPIE*, volume 3312, pages 150–161, 1997.

16. Nikos Paragios and Rachid Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3), March 2000.
17. Theodosios Pavlidis. Review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7(2):243–258, April 1978.
18. Silvia Pfeiffer, Rainer Lienhart, and Wolfgang Effelsberg. Scene determination based on video and audio features. In *IEEE Conference on Multimedia Computing and Systems*, 1999.
19. Brian Scassellati, Sophoclis Alexopoulos, and Myron Flickner. Retrieving images by 2d shape: a comparison of computation methods with human perceptual judgments. In Wayne Niblack and Ramesh C. Jain, editors, *Proceedings of SPIE, Storage and Retrieval for Image and Video Databases II*, volume 2185, pages 2–14, 1994.
20. Savitha Srinivasan, Dulce Ponceleon, Arnon Amir, and Dragutin Petkovic. What is in that video anyway?: In search of better browsing. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1999.
21. Michael D. Tarr and Heinrich H. Bülthoff, editors. *Object Recognition in Man, Monkey, and Machine*. MIT Press, Cambrigde, MA, 1998.
22. Shimon Ullman. *High-level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996.
23. Howard D. Wactlar. Informedia – search and summarization in the video medium. In *Proceedings of Imagina*, 2000.