Motion Reconstruction using Moments Analysis

Lourena Rocha Luiz Velho Paulo Cezar P. Carvalho
IMPA–Instituto Nacional de Matemática Pura e Aplicada
Estrada Dona Castorina, 110, 22460 - Rio de Janeiro, RJ, Brasil
{lourena, lvelho, pcezar}@visgraf.impa.br

Abstract

This paper presents a method for motion reconstruction. The approach is based on image moments and the bsp-tree data structure. We use invariant properties of these moments to construct a bsp-tree and determine a set of ellipses that aproximates the object's shape in each frame. These ellipsoidal structures are matched, which allow to track the object. Motion is represented by the geometric transformations between the sets of ellipses and stored in the hierarchical structure. Motion reconstruction is done by interpolating these transformations.

1. Introduction

Motion carries a lot of information about spatiotemporal relationships between image objects. Its reconstruction play a very important role in many areas such as video processing, compression and animation. Usually, in video processing it can be used for general temporal filtering of video. In video compression, it is usefull for the removal of temporal video redundancy; in an ideal situation, only the first image and the subsequent motion need to be transmitted. Also, motion reconstruction provides smooth animations by interpolating keyframes.

Recent works in reconstruction of object motion from a sequence of video images, deal with human and general 3D figure motion [5, 17, 19].

A fundamental step in motion analysis is the ability to perform object tracking. For the past years, several approaches for tracking have been developed. Common ones are based on features [22], optical flow [4, 3], active contours [12, 21] and shape models [18, 15]. Some drawbacks suffered by those methods range from low-level problem characterization to manual initialization and expensiveness.

Usually, techniques to track motion in 3D are based on tracking of 2D data [5, 17], for example, 2D correspondences that identify the projection of the 3D figure in each frame of a video sequence.

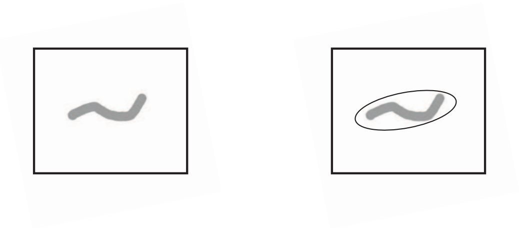
Based on this observation, we propose a method to reconstruct motion of binary objects. It is assumed that the motion is smooth. Our approach is based on *image moments*. Moment invariants allow us to infer the *ellipse of inertia*, i.e., the best fitting ellipse for a target 2D object [16, 13]. Since an image is the projection of a 3D scene, we can extend this property to get the best fitting ellipsoid for the target 3D object. In [2], ellipsoids are used to approximate the surface of a 3D object from volumetric data. We work only with 2D information and use image moments to perform structuring of dynamic objects. In this context, moments deal well with noise and uncertainty.

The algorithm is divided in three main stages. In the first stage, we use moments to construct a *bsp*-tree [9] for each frame of the sequence. This hierarchical representation approximates the object by 2^k ellipses at each level k of the tree. In the second stage, we use the hierarchical structure of a *bsp*-tree to track the object from frame to frame. Tracking allow us to compute geometric transformations between the sets of ellipses. Motion is represented by those transformations and, in the last stage, motion reconstruction is done by interpolating them.

This paper is organized in six sections including this introduction. Section 2 defines image moment and discusses some properties of them. Section 3 describes the mathematical theory developed by Alexa in [1], which let us perform the interpolation of geometric transformations. Section 4 is divided in tree subsections. In 4.1 the approximation of an object by ellipses is explained. In 4.2 we show how the hierarchical structure is used for tracking, and the subsection 4.3 describes the procedure that computes the interpolation between the structures. Results and applications are discussed in Section 5 and finally, Section 6 concludes the paper and give directions of future work.

2. Moments

Image moments and moment-based invariants have been used in pattern recognition and shape analysis for the past 40 years [11, 6, 25]. That is because the representation of



(a) (b)

Figure 1. Approximation based on image moments: (a) source image, (b) ellipse of inertia

image shape features is an active field of research and image moment functions have been demonstrated to be efficient shape descriptors [7, 8, 14].

The $(p+q)$th order two-dimensional geometric moments of a image intensity function $f(x, y)$ are defined as:

$$m_{pq} = \int\int_{\xi} x^p y^q f(x, y) dx dy, \tag{1}$$

p, q = 0, 1, 2 . . . , where ξ is the region of the pixel space in which $f(x, y)$ is defined.

Since the gray-level images can be thresholded to segment the object from the background, we will deal with binary images (*silhouette images*), in this paper. The pixels on the object region are assigned a value 1, and those on the background region are assigned a value 0. In this case, the region ξ consists of only those pixels which correspond to points on the object and have a value 1. Thus, we will use the *Silhouette Moments* [16], which constitute a variation of the geometric moments, and are expressed as:

$$m_{pq} = \int\int_{\xi} x^p y^q dx dy \tag{2}$$

p, q = 0, 1, 2 . . . .

In our case $f(x, y)$ is a digital image, thus the double integral in (1) and (2) must be replaced by a summation. Then, for silhouette images we have:

$$m_{pq} = \sum_{\xi} x^p y^q \tag{3}$$

where the summation extends over all the elements in ξ, i.e., all the "black" pixels in the image.

From Eq. (3) we see that m_{00} represents the area of the object, i.e., the number of black pixels. The object centroid c can be calculated combining m_{00} with the silhouette moments of first-order m_{01} and m_{10}:

$$x_c = \frac{m_{10}}{m_{00}}, y_c = \frac{m_{01}}{m_{00}} \tag{4}$$

With this framework we can design shape features or measurements that are invariant to certain affine transformations [1].

The moments computed with respect to the object centroid are called *central moments* and are defined as:

$$\mu_{pq} = \sum_{A} (x - x_c)^p (y - y_c)^q. \tag{5}$$

These moments are invariants to translation.

A convenient representation for the overall structure of the object (size, position and orientation) is provided by its *ellipse of inertia* (Fig. 1). This ellipse is centered at the centroid c of the object. Its axes are the lines passing through the centroid for which the second-order central moments about such lines are maximum and minimum respectively; it can be shown that these lines are perpendicular to each other and correspond to eigenvectors of the covariance matrix of the object (i.e., the matrix whose entries are the second-order central moments with respect to axis going through the centroid).

The angle θ corresponding to the orientation of the major axis and the lengths (w, l) of the semi-axes can be expressed in terms of the zeroth-, first- and second-order central moments, as follows [16]:

$$\begin{aligned} w &= 2\sqrt{\frac{I_1}{\mu_{00}}}, \\ l &= 2\sqrt{\frac{I_2}{\mu_{00}}}, \\ \theta &= \frac{1}{2} tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \end{aligned} \tag{6}$$

1 We remark that, although we are discussing binary objects, it is possible to generalize all the following relations and results for gray-level objects.

where I_1, I_2 are defined as:

$$\begin{aligned} I_1 &= \frac{(\mu_{20}+\mu_{02}) + [(\mu_{20}-\mu_{02})^2 + 4\mu_{11}^2]^{\frac{1}{2}}}{2}, \quad (7) \\ I_2 &= \frac{(\mu_{20}+\mu_{02}) - [(\mu_{20}-\mu_{02})^2 + 4\mu_{11}^2]^{\frac{1}{2}}}{2}. \end{aligned}$$

Therefore, an ellipse of inertia can characterize the fundamental shape features, and also represents the two-dimensional position and orientation on an object. To provide a better visualization, in this paper we employ enclosing elements with dimensions ($2w$, $2l$).

3. Linear Combination of Transformations

In this section, we will describe briefly the approach developed by Alexa in [1], regarding geometric transformations. In that work, he derives a definition of scalar product and a commutative addition of transformations, based on their matrix representations. Those operations, together, allow the linear combination of transformations. We will use this mathematical tool to interpolate our hierarchical structure, as we will describe in 4.3.

Geometric transformations are fundamental in computer graphics, and are represented as square real matrices. This representation provides an easy way to manipulate and implement the operations between transformations.

Interpolation of transformations is necessary for the representation and reconstruction of motion, and the common way to accomplish this uses matrix factorization [23, 24], but this approach has some drawbacks. The main one is that standard matrix product is not commutative.

Alexa solved this problem by defining the operations $\odot$ and $\oplus$ – *scalar product* and *commutative addition* – of transformations, respectively.

Considering a transformation T, its matrix representation M and a scalar α, the scalar product $\odot$ is defined as:

$$\alpha \odot T = T^{\alpha},\ i.\ e,\ \alpha \odot M = M^{\alpha}. \quad (8)$$

If M has no negative real eigenvalues, $\alpha \odot T$ is well defined and the following properties hold:

- $0 \odot M = M^0 = I$;
- $1 \odot M = M^1 = M$;
- $M^r \odot M^s = M^{(r+s)} = M^s \odot M^r$.

Let A, B be two square real matrices of the same dimension. The addition $A \oplus B$ is defined as:

$$A \oplus B = \lim_{n\to\infty} \left(A^{1/n} B^{1/n}\right)^n. \quad (9)$$

If A and B have primary roots [10], the limit above exists, $\oplus$ is commutative, $A \oplus B$ is real if A and B are real, and the following properties hold:

- $A \oplus B = AB \Longleftrightarrow AB = BA$;
- $det(A \oplus B) = det(A)det(B)$;
- since A commutes with A^{-1}, the inverse of A with respect to $\oplus$ is the standard matrix (product) inverse A^{-1}.

The operations $\odot$ and $\oplus$ can be computed using the matrix exponential and logarithm [10]. They are expressed as:

$$r \odot A = e^{r\ logA} \quad (10)$$

and

$$A \oplus B = e^{(logA\ +\ logB)}. \quad (11)$$

Using these equations, we are able to perform a linear combination of an arbitrary number of transformations T_i with weights w_i:

$$\bigoplus_i (w_i \odot T_i) = e^{\sum_i (w_i\ \cdot\ logT_i)}. \quad (12)$$

We will use the formulation in Equation 12 to interpolate the hierarchical structure which constains the informations about the object motion.

4. The Method

We use a hierarchical *bsp*-tree to approximate the target object by a set of ellipses of inertia in each frame [20]. Each ellipse from the current frame is matched into an ellipse in the consecutive frame. Thus, we are able to track a particular part of the object troughout the frame sequence. We compute a geometric transformation between the correspondent ellipses, and finally achieve the representation and reconstruction of the motion by interpolating from one transformation to another.

4.1. Hierarchical Structure

For each frame of the image sequence a *bsp*-tree is constructed. Each node in the tree represents an ellipse of inertia that approximates a part of the target object. Level k of the tree ($k = 0$ for the root) has 2^k nodes, corresponding to a set of 2^k ellipses that represent the object at that level. The *bsp*-tree is assumed to have the same number of levels for all frames of the image sequence.

Each node contains the following parameters:

- μ_{00} - zeroth-order central moment;
- μ_{11}, μ_{02}, μ_{20} - second-order central moments;
- $c = (x_c, y_c)$ - object centroid (ellipse's center);
- w, l - lengths of the ellipse's major and minor semi-axis, respectively;
- θ - orientation angle of ellipse's major axis;
- d - direction of the ellipse's minor axis and bsp's partition line;
- M - matrix representation of the geometric transformation between this ellipse and its correspondent in the previous frame.

The parameters above are computed according to Eq. (4), Eq. (5) and Eq. (6), with exception of direction d, whose definition is:

$$d = (\cos(\theta + \frac{\pi}{2}), \sin(\theta + \frac{\pi}{2})), \tag{13}$$

with θ as in (6).

Given a frame i of the sequence and the number k of levels of the tree, the procedure that constructs the approximating *bsp*-tree consists of the following steps:

1 - Compute μ_{00}, μ_{11}, μ_{02}, μ_{20}, c, w, l, θ, d and get the ellipse of center c and minor axis with direction d. Place this information at the root node of the tree. (Figure 2(a))

2 - Subdivide the image in direction d and add two children nodes to the root, each one containing the information corresponding to one sub-image.(Figure 2(b))

3 - For each children node apply recursively the algorithm (steps 1 and 2) until reaching level k.

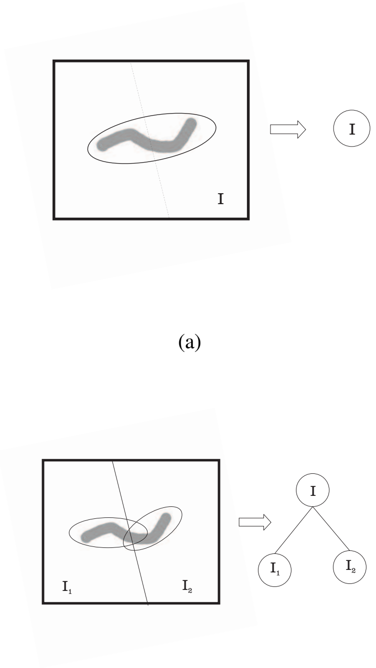
Thus, after applying this procedure for each image we will get a new sequence of frames where the object structure is represented by a set of ellipses.

4.2. Tracking

Let bsp_j be the tree at frame j and d_{ij} the bsp's partition line for each node i of bsp_j.

To guarantee the correspondence between ellipses in consecutive frames, it is necessary to maintain the consistency of the successive directions d_{ij}, according to the orientation angle θ calculated in each frame (Eq. 6).

If $\langle d_{ij}, d_{i(j+1)} \rangle \leq 0$, the correct measure for θ is $\theta = 180^0 + \theta$. Without this coherence test, one can get wrong matches. Thus, $bsp_{(j+1)}$ depends on bsp_j and the previous algorithm becomes:



(a)

(b)

Figure 2. Construction of the *bsp*-tree: (a) level 0, (b) level 1

1 - Construct bsp_0 for frame 0 as described in the previous subsection, and let d_{00} be the bsp partition line at the root node.

To construct the bsp_1 for frame 1 apply the following steps:

2 - Compute μ_{00}, μ_{11}, μ_{02}, μ_{20}, c, w, l, θ, d_{01} and get the ellipse of center c and minor axis with direction d_{01}. Place this information at the root node of the bsp_1.

3 - Verify if direction d_{01} is compatible with direction d_{00},i.e., if $\langle d_{00}, d_{01} \rangle > 0$. Otherwise, set $\theta = (180^0 + \theta)$ and compute d_{01} again.

4 - Compute the transformation between the ellipse at the root node of bsp_0 and the one at the root node of bsp_1. Store the matrix M which represents this geometric transformation.

5 - Add two children nodes to the bsp_1 root node, and for each one, apply recursively the algorithm (steps 2, 3 and 4) until reaching level k.

For each pair of consecutive frames apply the algorithm above.

4.3. Interpolation

Object motion is represented hierarchically in the *bsp*-tree structures that were computed and matched for all frames. To reconstruct this motion, we interpolate the transformations contained in those structures and compute new ones containing the interpolated parameters.

Given bsp_j and bsp_{j+1} *bsp*-trees of two consecutive frames in the sequence, and a real number $0 < exp < 1$, the pseudo-code of the procedure that constructs a new tree between the consecutive ones is:

Algorithm 1 BSP = Interp(bsp_j, bsp_{j+1}, exp)

if bsp_j $!=$ NULL and bsp_{j+1} $!=$ NULL **then**
  BSP $\Leftarrow$ NewNode()
  InterpNode(BSP, bsp_j, bsp_{j+1}, exp)
  BSP$\rightarrow$child1 = Interp($\text{bsp}_j\rightarrow$child1, $\text{bsp}_{j+1}\rightarrow$child1, exp)
  BSP$\rightarrow$child2 = Interp($\text{bsp}_j\rightarrow$child2, $\text{bsp}_{j+1}\rightarrow$child2, exp)
  return BSP
else
  return NULL
end if

The procedure `InterpNode` interpolates the parameters stored at two correspondent nodes, and it is given by:

Algorithm 2 InterpNode(BSP, bsp_j, bsp_{j+1}, exp)

BSP$\rightarrow$M = $(\text{bsp}_{j+1}\rightarrow\text{M})^{exp}$
BSP$\rightarrow$c = (BSP$\rightarrow$M) $\cdot$ ($\text{bsp}_j\rightarrow$c)
A, B $\Leftarrow$ extremities of ellipse major axis in bsp_j
a, b $\Leftarrow$ extremities of ellipse minor axis in bsp_j
A' = (BSP$\rightarrow$M) $\cdot$ A
B' = (BSP$\rightarrow$M) $\cdot$ B
a' = (BSP$\rightarrow$M) $\cdot$ a
b' = (BSP$\rightarrow$M) $\cdot$ b
BSP$\rightarrow$w = $\| B' - A' \| /2.0$
BSP$\rightarrow$l = $\| b' - a' \| /2.0$
BSP$\rightarrow\theta$ = angle($\overrightarrow{A'B'}$, x-axis)

To get all the benefits of this motion interpolation, it is necessary that the object motion have been well represented at the ellipses set.

5. Results and Applications

In this section we show some results in order to demonstrate the structuring and tracking algorithms. In addition, we discuss how to apply the algorithm to perform object reconstruction.

5.1. Ellipse Fitting and Tracking

Figure 3 shows the structuring algorithm for a "S" shape. More examples can be seen at [20].

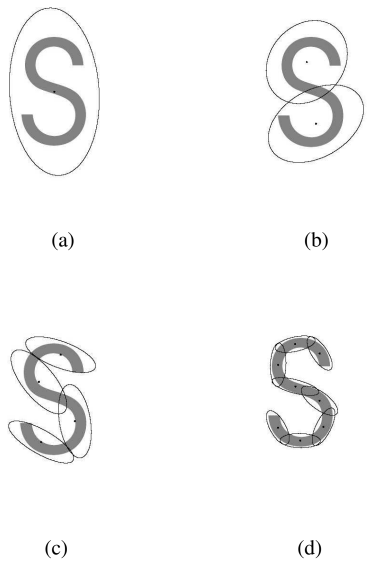


Figure 3. Object fitting at levels 0, 1, 2 and 3

Figures 4 and 5 illustrate the tracking algorithm. In Figure 4, all ellipses at level 3 for a deforming object are shown. One can observe that the ellipses capture the overall movement of the object. In Figure 5 we show only the ellipse for a level 2 decomposition, in order to demonstrate that the algorithm is capable of maintaining an appropriate correspondence between the set of ellipses for each frame.

5.2. Object Reconstruction

Assuming that input images in the sequence are binary, that they contain just one target object, and the motion is smooth, as in our algorithm, we are interested in reconstructing a image I^* between two consecutive images I_i and I_{i+1} of the input sequence.

The first step is to construct the *bsp*-tree for each image and match them. For this, we need to choose the best number k of fitting ellipses to represent the target object. (Re-

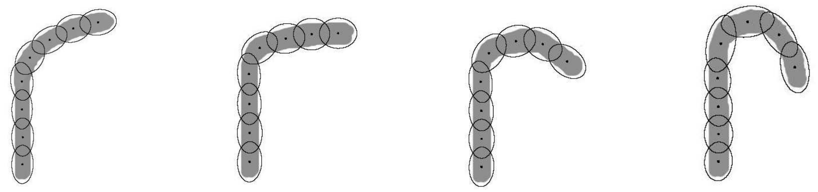


Figure 4. **Tracking object at level 3, showing all ellipses at that level**

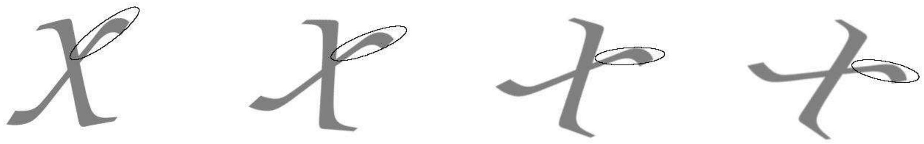


Figure 5. **Tracking object at level 2 (only one ellipse is shown)**

member that all *bsp*-trees have the same quantity of levels, and level k contains 2^k nodes). Next, the geometric transformations between I_i and I_{i+1} are calculated and stored at the *bsp*-tree of I_{i+1}.

Supppose that we want to interpolate exactly one frame between I_i and I_{i+1}. This means, for example, that we want to apply to the scene in I_i "half" the transformations that takes I_i into I_{i+1}. In this case, we use exp=0.5 at the interpolation algorithm.

For each leaf node of the interpolated tree we identify its correspondent sub-image. For each pixel location in the sub-image, the inverse of the transformation stored at the correspondent leaf node is applied, leading to a pixel location in I_i. The pixel at the destination receives the same color as the pixel in the origin.

To reconstruct rigid motion, just the *bsp*-tree root node (level $k = 0$) is required, since motion in each object's piece is the same. This means that the object is represented by only one ellipse. Figure 6 shows two consecutive frames and a intermediate one, which was interpolated with exp = 0.5. The ellipses representing the object in each frame are shown, as well. In addition to rotation and translation, this example contains a slight scale.

We need to pay more attention to non-rigid motion, since image discontinuity may occur (Fig. 7). This happens when we use information stored at level k of the *bsp*-tree, with $k > 0$, to perform the reconstruction. The reason is that, at this time, the algorithm handles with motion in each sub-image individually.

Even thus, using level k, $k > 0$, provides a better reconstruction for non-rigid motion (Fig. 8). Although discontinuity appears, it decreases when k increases (Fig. 9).

6. Conclusion and Future Works

In this paper we have proposed a method for reconstruction of motion in an image sequence. In addition, the presented technique performs structure and tracking of moving objects. The algorithm uses image moments to build a hierarchical *bsp*-tree structure, which approximates the object by a set of ellipses at different levels. Tracking is performed by exploiting structural and temporal coherence of the representation. Geometric transformations between those ellipsoidal structures are computed and motion reconstruction is achieved by interpolating them.

Some problems that we did not address in this paper are occlusion, tracking of multiple objects and motion discontinuities. Future work will go in these directions. Also, we intend to handle images discontinuities in non-rigid motion and to apply the algorithm to track real objects in video sequences.

Another possible application consists of transferring the captured movement of a given object to another. For instance, this could be used to produce new character animation based on the behavior of an existing animation.

We also would like to extend our approach to 3D space. We intend to use ellipsoids instead ellipses, getting the 3D approximation of the object.

(a) (b)

Figure 6. (a) Grey image was reconstructed using exp = 0.5. (b) Ellipses representing the object in each frame.

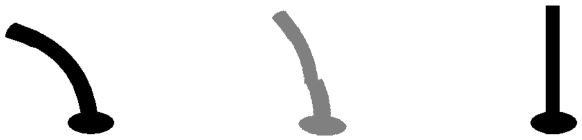


Figure 7. Image discontinuity. Gray image was interpolated using exp = 0.5 and *bsp*-tree level k = 1.

Acknowledgments

The first author would like to thank José Luiz Luz for discussions contributing the ideas in this paper and for help in the implementation.

This research has been developed in the VISGRAF Laboratory at IMPA. VISGRAF Laboratory is sponsored by CNPq, FAPERJ, FINEP and IBM Brazil.

References

[1] M. Alexa. Linear combination of transformations. In *Proceedings of the 29th Annual Conference on Computer graphics and interactive techniques*, pages 380–387. ACM Press, 2002.

[2] F. Banegas, M. Jaeger, D. Michelucci, and M. Roelens. The ellipsoidal skeleton in medical applications. *Sixth Solid Modeling*, June 4-8 2001.

[3] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkit. Performance of optical flow techniques. *Computer Vision and Pattern Recognition*, pages 236–242, 1992.

[4] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proc. Third IEEE Conference on Face and Gesture Recognition*, Nara, Japan, April 1998.

[5] D. E. Difranco, T. J. Cham, and J. M. Rehg. Reconstruction of 3d figure motion from 2d correspondences. In *IEEE Int. Conference Computer Vision and Pattern Recognition*, Kauai, Hawaii, November 2001.

[6] S.A. Dudani, K.J. Breeding, and R.B. McGhee. Aircraft identification by moment invariants. *IEEE Transactions on Computers*, 26(1):39–46, January 1977.

[7] J. Flusser and T. Suk. Pattern recognition by affine moment invariants. *Pattern Recognition*, 26(1):167–174, 1993.

[8] J. Flusser and T. Suk. Affine momemt invariants: A new tool for character recognition. *Pattern Recognition Letters*, 15:433–436, 1994.

[9] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics*, 14:124–133, 1980.

[10] R. A. Horn and C. A. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[11] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions and Information Theory*, IT-8:179–187, Feb 1962.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Conference on Computer Vision*, (First):321–331, 1987.

[13] K.L. Lau, W.C. Siu, and N.F. Law. Best-fit ellipse concept from second order moments. In *IEEE International Conference on Information Technology and Applications, (ICITA2002)*, pages 167–75, Bathurst, Australia, November 2002.

[14] S. X Liao and M. Pawlak. On image analysis by moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(3):254–266, March 1996.

[15] D. Metaxas. Shape representation and nonrigid motion tracking using deformable superquadrics. *Geometric Methods in Computer Vision*, pages 12–20, July 1991.

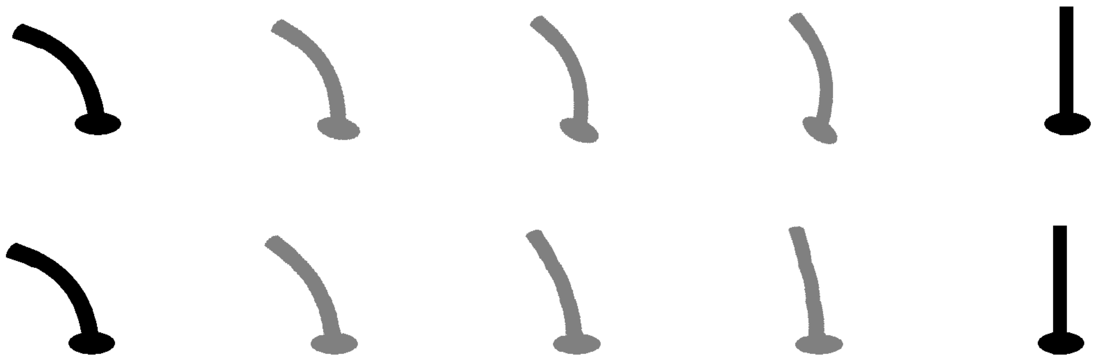


Figure 8. Although for non-rigid motion the interpolated images are discontinuous when $k > 0$, it provides a better motion reconstruction. Images interpolated with exp = 0.25, 0.50 and 0.75. First row: *bsp*-tree level = 0. Second row: *bsp*-tree level = 2.

(a) (b)

Figure 9. Image discontinuity decreases when k increases. (a) $k = 1$. (b) $k = 2$.

[16] R. Mukundan and K. R. Ramakrishnan. *Moment Functions in Image Analysis: Theory and Applications*. World Scientific World Scientific Publishing Co Pte Ltd., Singapore, 1998 (September).

[17] M. J. Park, M. G. Choi, and S. Y Shin. Human motion reconstruction from inter-frame feature correspondences of a single video stream using a motion library. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 113–120. ACM Press, 2002.

[18] A. Pentland. Modal descriptions for recognition and tracking. *IAPR Workshop on Machine Vision Applications*, pages 435–444, december 1992.

[19] Z. Popov and A. Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM Press/Addison-Wesley Publishing Co., 1999.

[20] L. Rocha, L. Velho, and P. C. Carvalho. Image moments-based structuring and tracking of objects. In *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*, pages 99–105. IEEE Computer Society, 2002.

[21] S. Sclaroff and J. Isidoro. Active blobs: region-based, deformable appearance models. Technical Report 2-3, 2003.

[22] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, Jun 1994.

[23] K. Shoemake. Animating Rotation with Quaternion Curves. In *"Proceedings of" SIGGRAPH 1985*, volume 19, pages 245–254. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 22–26 1985.

[24] K. Shoemake. *Graphics Gems II*, chapter Quaternions and 4x4 matrices, pages 351–354. Academic Press, Boston, 1991.

[25] C. H. Teh and R. T. Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):496–513, July 1988.