J Neurophysiol 98: 1775–1790, 2007.
First published July 11, 2007; doi:10.1152/jn.00739.2006.

Analyzing Octopus Movements Using Three-Dimensional Reconstruction

Yoram Yekutieli,[1,2,3] Rea Mitelman,[1,2] Binyamin Hochner,[1,2] and Tamar Flash[3]

[1]*Department of Neurobiology and* [2]*Interdisciplinary Center for Neural Computation, Hebrew University, Jerusalem; and* [3]*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel*

Submitted 18 July 2006; accepted in final form 8 July 2007

Yekutieli Y, Mitelman R, Hochner B, Flash T. Analyzing octopus movements using three-dimensional reconstruction. *J Neurophysiol* 98: 1775–1790, 2007. First published July 11, 2007; doi:10.1152/jn.00739.2006. Octopus arms, as well as other muscular hydrostats, are characterized by a very large number of degrees of freedom and a rich motion repertoire. Over the years, several attempts have been made to elucidate the interplay between the biomechanics of these organs and their control systems. Recent developments in electrophysiological recordings from both the arms and brains of behaving octopuses mark significant progress in this direction. The next stage is relating these recordings to the octopus arm movements, which requires an accurate and reliable method of movement description and analysis. Here we describe a semiautomatic computerized system for 3D reconstruction of an octopus arm during motion. It consists of two digital video cameras and a PC computer running custom-made software. The system overcomes the difficulty of extracting the motion of smooth, nonrigid objects in poor viewing conditions. Some of the trouble is explained by the problem of light refraction in recording underwater motion. Here we use both experiments and simulations to analyze the refraction problem and show that accurate reconstruction is possible. We have used this system successfully to reconstruct different types of octopus arm movements, such as reaching and bend initiation movements. Our system is noninvasive and does not require attaching any artificial markers to the octopus arm. It may therefore be of more general use in reconstructing other nonrigid, elongated objects in motion.

INTRODUCTION

The flexible octopus arm is an amazing organ used in various motor tasks such as locomotion, food gathering, hunting, and sophisticated object manipulation (Fiorito et al. 1990; Mather 1998; Wells and Wells 1957). The arm senses its environment using tactile and chemosensory organs and passes this information to the brain (Graziadei 1971; Rowell 1966). Octopus arms, as well as squid tentacles, elephant trunks, and vertebrate tongues, belong to a group of organs termed *muscular hydrostats*. All these structures lack a rigid skeleton and their structural support and force transmission are achieved only through their musculature. Although the biomechanical principles of muscular hydrostats are well understood (Kier 1992; Kier and Smith 1985; Kier and Thompson 2003), relatively little is known about the neural control of movement in these structures.

Our research group is conducting a large-scale research project investigating octopus motor control. To date, two stereotypical movements have been extensively studied, the *reaching movement* (Fig. 1*A*; Gutfreund et al. 1996, 1998; Sumbre et al. 2001; Yekutieli et al. 2002, 2005a,b) and the *fetching movement* (Fig. 1*B*; Sumbre et al. 2001, 2005, 2006). These movements were studied by analyzing the position and velocity of a specific bend point (or points) along the arm. The kinematics, together with electromyographic recordings and detailed biomechanical simulations, revealed some common principles of octopus arm motor control. In both reaching and fetching movements the complexity associated with the control of the flexible arm is reduced by coupling among different degrees of freedom. Despite the significance of these findings, we believe that the insights—gained through kinematic description of the movement based on the movement of a single or a few points along the arm—are rapidly reaching their limits.

New tools and methods are therefore required for analyzing the movements of the entire arm in their full complexity. For example, a significant part of the octopus arm is frequently used as a manipulator, making it difficult or impossible to describe by the kinematics of a small number of points along the arm. Any part of the arm can catch and grip an object by using the suckers and wrapping the arm around the object (Fig. 1*B*). During foraging the whole arm searches and senses the environment (Fig. 1*C*). Zullo et al. (2005) recently developed a system for electrophysiological recordings in the brains of behaving octopuses. These recordings should be interpreted in conjunction with a detailed description of the three-dimensional (3D) movements of one or more arms.

Tracking octopus arms

An initial step toward a complete 3D reconstruction of the octopus arm movements involves the reliable acquisition of motion data. Here we focus only on tracking the arm in video sequences and describe the difficulties associated with such tracking. For each video frame of a given motion we need to track the arm movement by successfully segmenting it from the background. The nonrigid nature of the octopus arm poses problems in automatically achieving this segmentation. In articulated structures (such as the human arm or the exoskeleton of arthropods) simple kinematic relationships among the moving segments can be used to track and reconstruct the movements (Aggarwal and Cai 1999; Gavrila 1996; Jennings 1999; Zakotnik et al. 2004). In contrast, the octopus arm undergoes not only rigid body transformations but also changes its shape, making motion tracking task much more difficult. Moreover, in images of natural octopus movements, occlusions of one arm by itself, by other arms, or by the octopus' body are frequent (Fig. 2). Other difficulties arise from the physics of the scene, e.g., shadows from the upper surface of the water may obscure the shape of the octopus arm.

Address for reprint requests and other correspondence: T. Flash, The Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, POB 26, Rehovot 76100, Israel (E-mail: tamar.flash@weizmann.ac.il).

The costs of publication of this article were defrayed in part by the payment of page charges. The article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. Section 1734 solely to indicate this fact.

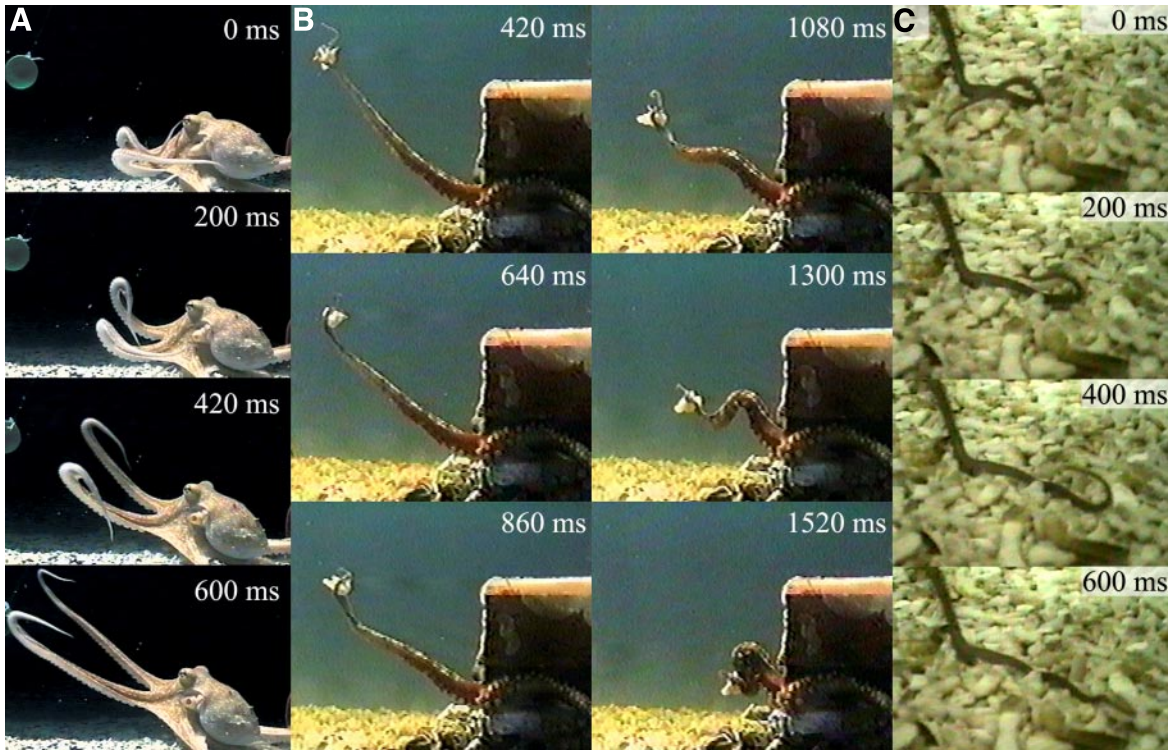

FIG. 1. Examples of octopus movements. *A*: 4 images from a video sequence of an octopus reaching with 2 arms toward a small plastic disc (*top left*). A bend is formed in the arms and is then propagated distally along the arms until both are straight. Bases of the arms are rotated during movement to orient the arms toward the target. Suckers along the arms are kept facing outward, preparing the arm to grab anything it hits anywhere along the arm. *B*: 6 images from a video sequence of a fetching movement, where an octopus catches a piece of food and brings it back to its mouth. Arm forms 3 pseudojoints: the distal one near the food, the middle one at half the distance between the first joint and the base of the arm and a third joint at the base of the arm. *C*: 4 images from a video sequence of an octopus arm searching around the aquarium. Bend propagation is apparent as a submovement in this sequence. Bend initiations and sideways movements of the tip are also common in searching movements (not shown here).

In a tracking application, it may be useful to artificially mark the object that is to be tracked by painting it with some color or by attaching some markers to key points on the object's surface (Bodenheimer et al. 1997; Hughes and Kelly 1996; Mazzoni et al. 2005). This is very difficult to do with the octopus arm, for two reasons. First, the octopus can rapidly change the color and even the texture of its skin, lowering the contrast of artificial markers (Moynihan 1975; Packard and Sanders 1971). Second, octopuses try to remove items attached to their skin and, because their arms are very dexterous, they are usually able to get rid of any such nuisances quite quickly.

FIG. 2. Occlusion of octopus arms. *Left* and *right* images taken from a video sequence of an octopus raising its arms (×2 zoom). Segmentation of the 2 arms in the *left view* is simpler than in the *right view*, where one arm occludes the other. Automatic segmentation may fail in the *right view*. Humans, however, can easily deal with such common problems and successfully segment the arm from the background even in much more difficult cases.

At the current stage of the system's development, we decided to circumvent the problems of automatic tracking and segmentation by relying on the human cognitive ability for performing these processing steps. A separate system for automatic segmentation and tracking of the octopus arm is under development (I. Zelman, M. Galun, A. Akselrod-Ballin, Y. Yekutieli, and T. Flash, unpublished observations).

Objectives

The aim of the research presented here is to construct a system capable of achieving accurate 3D kinematic description of a whole octopus arm in motion.

METHODS

First we enumerate the stages we have used on the way to reconstruction and then we elaborate separately on each stage in greater detail, focusing on the techniques and on the specific problems encountered when attempting to reconstruct the underwater movement of flexible, elongated shapes.

System description

The system we developed takes the following steps to achieve reconstruction (Fig. 3).

- Camera calibration
- Movement recording using two cameras in stereo configuration.
- For each video frame, the method involves:

1) Manual tracking of the contour of the arms from base to tip in the two views.

2) Automatic extraction of the two-dimensional (2D) midline (the backbone) of each contour.

3) Matching the two 2D midlines using epipolar geometry.

4) 3D reconstruction.

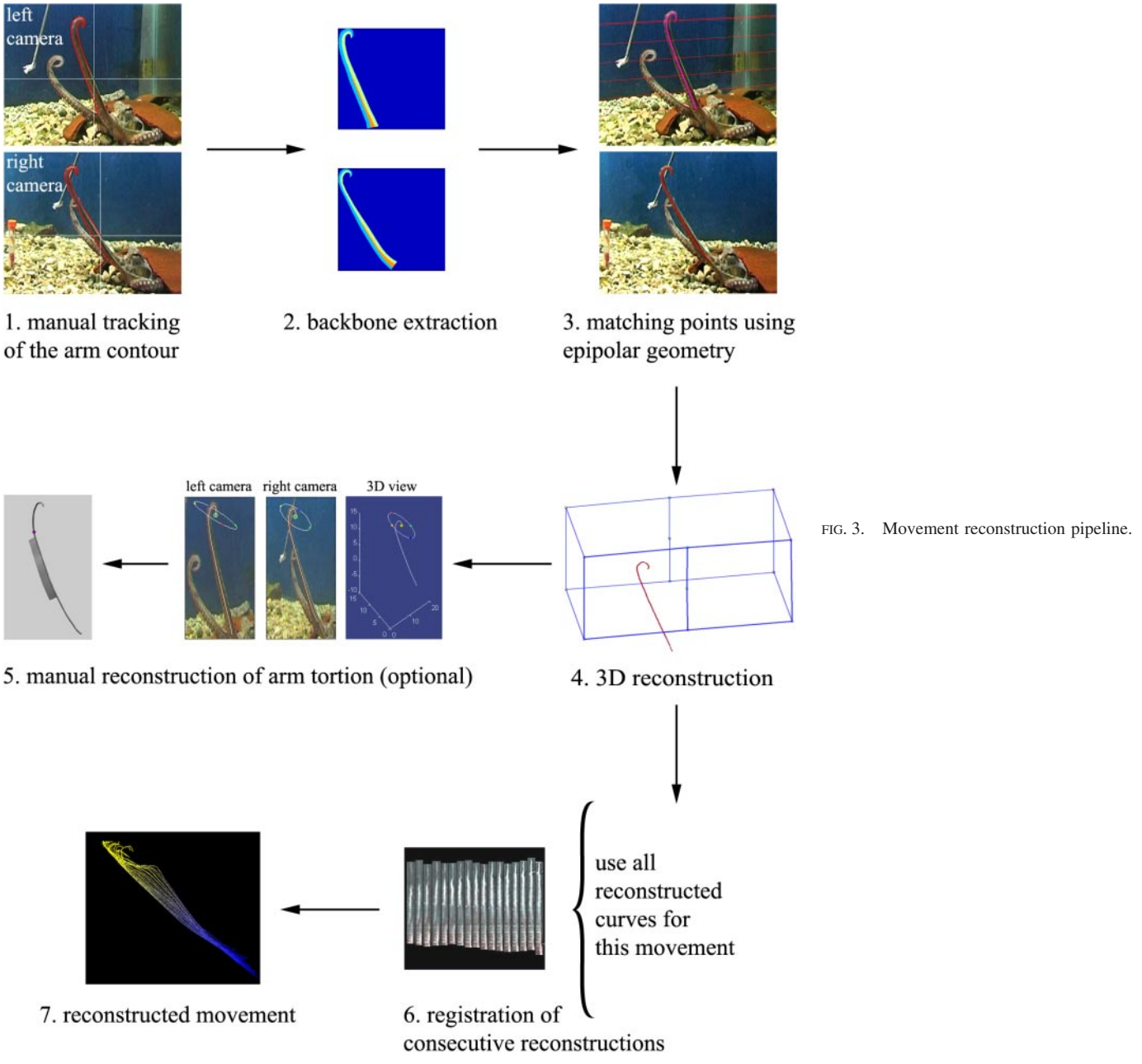

FIG. 3. Movement reconstruction pipeline.

5) Manual marking the orientation of the suckers in the two views to reconstruct the torsion of the arm in 3D (optional).

- Registration of consecutive reconstructions for the whole movement.

Physical setup

Viewing (and filming) an object in an aquarium raises the problem of optical refraction. The deformation of the image of the underwater object may degrade 3D reconstruction. According to Snell's law, when the angle of incidence is zero, there are no image distortions arising from light refraction. Therefore to minimize these distortions the two cameras should be positioned with their viewing angles normal to the aquarium facets or as close to normal as possible (for a comprehensive review on the effects of light refraction on the accuracy of camera calibration and reconstruction in underwater motion analysis see Kwon and Casebolt 2006).

We have tried two configurations, the *one-facet configuration*, where the two cameras face the same side of the aquarium, and the *two-facet configuration*, where the cameras face two different but adjacent facets. The latter has the lowest possible refraction error because both cameras are normal to the aquarium facet. Another advantage of the two-facet configuration is that the viewing angles of the two cameras are perpendicular to one another, reducing to a minimum the uncertainty of reconstruction (extracting depth information depends on the disparity found between the images). The images, however, are dramatically different and therefore an automatic registration (finding the correct correspondence) of the two images is very difficult. In the one-facet configuration, on the other hand, the images may be relatively similar, depending on the location of the two cameras. Putting the cameras as close as possible to each other reduces the refraction error but increases the uncertainty of reconstruction.

We mainly used the one-facet configuration with 25 to 50° between the principal rays of the cameras. A detailed analysis of the systematic error resulting from light refraction revealed that these errors are small in our setup (see RESULTS).

Data acquisition

Two video streams are recorded onto DV cassettes by synchronized digital cameras (Panasonic AG-DVC30E camcorder). The data are later downloaded to a standard PC computer using Adobe Premier (1.5) software and a FireWire connection (IEEE 1394). We use the PAL video system at 25 frames/s with image resolution of 720 × 576. Each frame is composed of two interlaced fields (half images that are composed either of the odd horizontal lines or the even horizontal lines). After retrieving the two fields, each is interpolated to a full-sized frame, achieving time resolution of 50 images/s.

Dedicated software that we developed using Matlab (version 6.5, The MathWorks) handles all image manipulation, marking, and analysis.

Camera model and calibration

We used a simple pinhole camera model commonly used in computer vision for perspective projection (Abdel-Aziz and Karara 1971; Cipolla and Giblin 2000; Roberts 1965). Each camera has 11 parameters that define the position and orientation of the camera, as well as optical characteristics, such as focal length, x–y ratio, and image center. Radial distortions are not accounted for by the pinhole model, but because they were small in our experimental setup, we could neglect such corrections.

The following notation is used throughout. Using homogeneous coordinates, a 3D point $\mathbf{q}$ is represented by $[X, Y, Z, 1]^T$ and a 2D point $\mathbf{p}$ by $[x, y, 1]^T$. A 2D line satisfying the equation $ax + by + c = 0$, is described by the vector that includes its parameters $[a, b, c]^T$.

A 3D point $\mathbf{q}$ is projected onto a 2D point $\mathbf{p}$ on the image plane by the projection equation

$$\mathbf{p} = \mathbf{M}\mathbf{q} \quad (1)$$

The camera projection matrix $\mathbf{M}$ is a 3 × 4 matrix that is defined up to a scale factor. The elements of $\mathbf{M}$ are the 11 parameters of the pinhole camera model.

Calibration is the process of determining these 11 parameters for each camera. First, both cameras record a 3D object with known geometry—the calibration frame. This object must have at least six points that are clearly visible and detectable at the two views. Then, the 2D position of each of these points is marked in each view. A least-square procedure estimates the camera parameters so that the viewed 2D points fit the known 3D geometry (Abdel-Aziz and Karara 1971; Cipolla and Giblin 2000). We use a 40 × 20 × 20-cm calibration frame, composed of thin metal rods connected by acrylic glass connectors. It has 15 LEDs (standard light-emitting diodes, 2.5 mm diameter) of three distinct colors, which are clearly visible and can be easily marked in two views (Fig. 11*A*). Marking is done manually with subpixel precision using a dedicated graphical user interface (GUI).

Before calibration, the relative position of the center of each LED should be known with submillimeter precision. We achieved this as follows. Initially we estimated the positions based on the 40 × 20 × 20-cm dimensions of the calibration frame. To check this estimation we measured the actual distances between each LED and its neighbors with Vernier calipers and compared the measured distance to the calculated distances of our initial estimate. This comparison gave an average error of 0.5 mm per measurement. Then, using a nonlinear optimization technique (Matlab's *fminsearch* function that uses the simplex search method; Lagarias et al. 1998) and our initial estimated position as its first guess, we found the set of LED positions that better fitted the measured distances (with an average error of 0.07 mm per measurement).

After calibration, it is possible to reconstruct the position of a 3D point from its projections, using the direct linear transform (DLT) procedure (Abdel-Aziz and Karara 1971; Gutfreund et al. 1996; Kwon 2007). The space that the calibration frame occupies is referred to as the calibrated space and points outside this space have higher reconstruction errors (Chen et al. 1994; Hinrichs and McLean 1995; Kwon 1999; Kwon and Casebolt 2006). The accuracy of our calibration and reconstruction is estimated below (see RESULTS).

Behavioral recording sessions

EXPERIMENTAL ANIMALS. Specimens of *Octopus vulgaris* were either caught on the Mediterranean shore by local fishermen or supplied by the Stazione Zoologica in Naples, Italy. The weight of the animals studied ranged from 200 to 700 g. The animals were maintained in 50 × 50 × 40-cm aquariums containing artificial seawater. The water was circulated continuously in a closed system and filtered by active carbon, mechanical, and biological filters. The animals were held at a temperature of 17°C on a 12-h light/dark cycle. A few days before an experiment, an octopus was moved to a larger aquarium (80 × 80 × 60 cm) kept at a water temperature of 18–20°C.

BEHAVIORAL TASK AND VIDEO RECORDING. To generate the reaching movements, a 2-cm-diameter green plastic disk (see Fig. 1*A*) was lowered into the water and moved slightly to attract the attention of the octopus. The octopus either extended one or more arms or swam toward the target. Every few trials, the animal was rewarded with a piece of crab meat tied to the target. Bend initiation movements were filmed either during spontaneous behavior or when a piece of meat was presented to the animal. Sometimes it was enough just to draw the attention of the animal to an object (such as forceps) near the surface of the water. The video recordings of animal behavior were searched

off-line for specific movement sequences (typically 0.5 to 1 s) that were then converted into a set of consecutive frames, ready for further analysis. We than marked the shape of one of the arms in the images from both cameras for the entire sequence, as described in the next section.

Manual tracking of arm contour

We developed a dedicated GUI to enable manual tracking of the contour of the arm in each view and throughout the movement. The GUI enables interactive zooming of the images to enhance the visibility of the arm contour and help the user in the decision where exactly to mark. A trained user can usually deal with poor image quality and ambiguous situations (as seen in Fig. 2) by studying the shape of the arm in time and using different zoom scales. The user marks the contour of the arm from the base of the arm, toward the arm tip and back toward the base. The actual marking is done by placing a piecewise linear curve along the contour. The user moves the mouse cursor freely and adds a point to the curve only when certain about the position. In this way we avoid the characteristic high-frequency jitter of freehand drawing. On completion, the marked curve is resampled to an equidistant, dense representation with 1-pixel distance between adjacent points (so the number of points is usually several hundred. Also note that the actual positions of the points are not rounded to the nearest integer values, avoiding unnecessary round-off errors). The same procedure is done for the two views and for every time step.

Next, the shape of every contour is processed automatically. The position of the tip of the arm is found (using the point of maximal curvature) and the number of intersections of the curve with itself is counted. This information is crucial in finding the midline of the arm, as described in the following sections. The contour is then processed to extract its backbone (2D midline) to later build a 3D midline representation of the octopus arm. The method is similar to the representation used by Chirikjian and Burdick (1994) to capture the major kinematic features of a hyperredundant robotic arm.

Finding the 2D midline of an arm

We define the midline of the octopus arm as starting at the base of the arm, going all the way to the tip and keeping an equal distance from the two sides of the arm contour. This midline is similar to the *skeleton* (or medial axis) of the shape, which is a well-known geometrical entity, a curve approximating the local symmetry axis of a shape (Golland and Grimson 2000). Blum (1967) described the skeleton using the metaphor of *grass fire*—a wave front is initiated from an outline curve and propagates at constant speed. The skeleton is the set of points where at least two wave fronts meet. Finding the midline is not identical to the problem of finding the skeleton of a shape. The difference is that the midline that we are interested in is composed of only one nonbranching curve, as opposed to the skeleton that can be composed of several branches.

The solution we use to find the midline is based on the grass-fire model and is described in detail in APPENDIX A. First the contour is divided into two sides from base to tip. Then two distinct waves are initiated from the two sides of the contour and are propagated at an equal speed inward (Fig. 4*A*). The set of points where the wave fronts collide is the midline (Fig. 4*B*). A further step of border following is then needed to convert the midline into an ordered set of points.

If the octopus arm forms a loop, the contour crosses itself and a straightforward implementation of the wave propagation solution is not sufficient. Instead, the arm shape is segmented into two parts and the wave propagation solution is implemented separately in each part. The resulting midlines of the two parts are combined together to form one continuous midline.

We now have two backbone curves, one for each view, ready for 3D reconstruction. The technique we use is based on finding a match between the points of one backbone curve to the points of the other, as described next.

The matching problem and epipolar geometry

Given two 2D points that are the projections of one 3D point and given the projection matrices, the 3D position of the source can be calculated. Because our data are composed of the midline curves extracted from images captured by two cameras, we need to find the correct matching between the points on the two views. This matching ensures that any pair of 2D points (a left point and a right point) originated from the same 3D point. A geometrical relation between the two views—the epipolar geometry (Fig. 5)—states that for any point on one view there exists a line on the other view where the matching point should be (Faugeras 1993; Hartley 1992). The epipolar geometry reduces the search for matching points from a 2D search (on the whole plane) to a 1D search on that line. The epipolar geometry is described by *Eqs. 2* and *3* in which $\mathbf{F}$ is a 3×3 matrix termed the *fundamental matrix*. $\mathbf{F}$ is constrained to have rank 2 (i.e., $\det \mathbf{F} = 0$) with only 7 degrees of freedom (df) and not 9 df

$$[x_1 y_1 1] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0 \quad (2)$$

or, compactly

$$\mathbf{p_1^T F p_2} = 0 \quad (3)$$

The left point $\mathbf{p_1}$ should lie on an epipolar line $\mathbf{l_1}$, i.e., $\mathbf{p_1^T l_1} = 0$, and the parametric representation of that line is defined by

$$\mathbf{l_1} = \mathbf{F p_2} \quad (4)$$

Similarly, *Eq. 3* leads to the definition of the other epipolar line $\mathbf{l_2}$

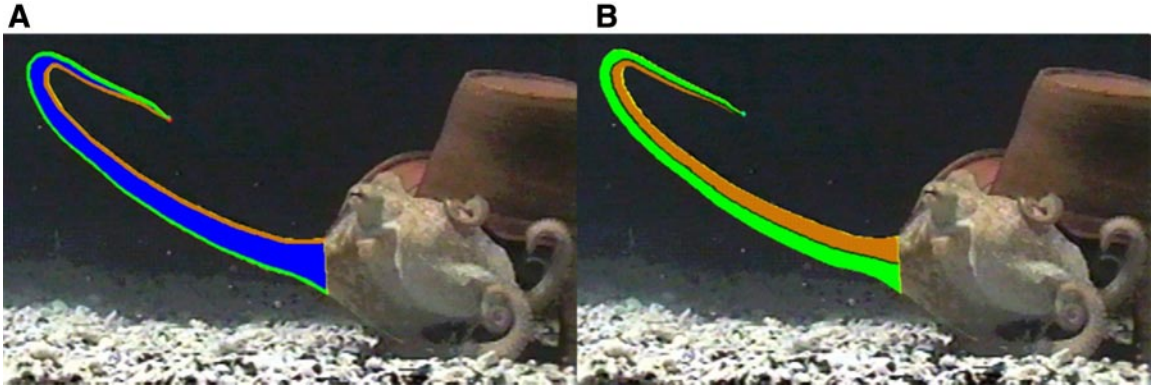

FIG. 4. Midline of the arm is found by the collision of 2 waves that propagate from the 2 sides of the contour inside the arm.

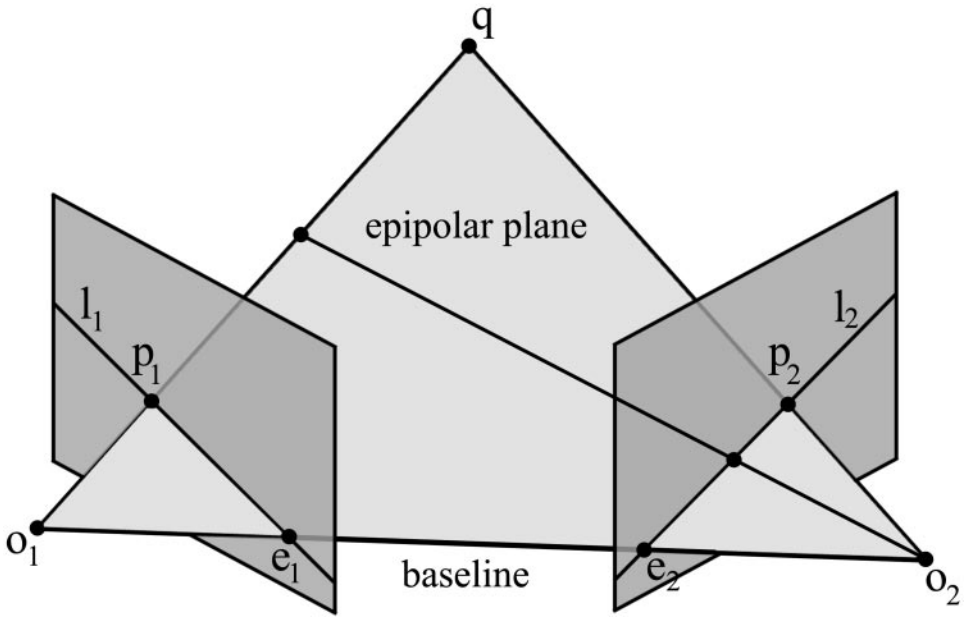

FIG. 5. Epipolar geometry. A 3D point $\mathbf{q}$ projects to a 2D point $\mathbf{p_1}$ on the left camera plane and to $\mathbf{p_2}$ on the right camera plane. *Optic centers* of the 2 cameras ($\mathbf{o_1}$, $\mathbf{o_2}$) and the point $\mathbf{q}$ create the *epipolar plane*. Intersections of this plane with each camera plane are the *epipolar lines* $\mathbf{l_1}$ and $\mathbf{l_2}$. *Epipoles* ($\mathbf{e_1}$, $\mathbf{e_2}$) are the projections of the optic center of each camera onto the other camera plane. Any point on the ray ($\mathbf{q}$, $\mathbf{o_1}$) projects to the right epipolar line $\mathbf{l_2}$.

$$\mathbf{l}_2^{\mathrm{T}} = \mathbf{p}_1^{\mathrm{T}}\mathbf{F} \tag{5}$$

In our case, because the projection matrices $\mathbf{M_1}$ and $\mathbf{M_2}$ are known (from the calibration), $\mathbf{F}$ can be derived algebraically (Xu and Zhang 1996)

$$\mathbf{F}^{\mathrm{T}} = [\mathbf{e}_2]_{\times}\mathbf{M}_2\mathbf{M}_1^{+} = [\mathbf{M}_2\mathbf{o}_1]_{\times}\mathbf{M}_2\mathbf{M}_1^{+} \tag{6}$$

where $[\,]_{\times}$ is a skew symmetric matrix used to perform a vector cross-product as a matrix multiplication, $\mathbf{e_2}$ is the epipole of the right camera (the intersection of all epipolar lines in this camera and also the projection of the optical center of the left camera onto the right camera: $\mathbf{M_2o_1}$), $\mathbf{M_1^+}$ is the pseudoinverse of $\mathbf{M_1}$, and $\mathbf{o_1}$ is the optical center of the left camera, which is the null space of $\mathbf{M_1}$ (defined by $\mathbf{M_1o_1} = 0$).

Taken together, the epipolar constraint and the midline representation are sufficient to match between points: any point on the midline of one view should have a matching point on the other view, at the intersection of the midline and the epipolar line (Fig. 3, step 3 and Fig. 8*D*).

However, as the angle between the midline and the epipolar line becomes very small, the error in calculating the intersection becomes large, and when the two lines are tangential, their intersection is not defined. Unfortunately, this situation is common in octopus movements and may occur several times along the midline of an arm (Fig. 6). In practice we would like to identify such cases before the matching is attempted. We cannot use the epipolar line as defined by *Eqs. 4* and *5* because these definitions assume that the matching is already known. Instead, we use the *self-epipolar line* of a point, which is the intersection of the epipolar plane with the image plane (Fig. 5): for a point $\mathbf{p_1}$ on the left image, its epipolar line on the right image is $\mathbf{l_2}$. Any point on $\mathbf{l_2}$ can be used to calculate its own epipolar line $\mathbf{l_1}$ in the left image. The point $\mathbf{p_1}$ must be on the line $\mathbf{l_1}$, so we term it the *self-epipolar line* of point $\mathbf{p_1}$. The angle between the backbone curve at the point $\mathbf{p_1}$ and the line $\mathbf{l_1}$ is then used as a measure of epipolar tangency.

Although accurate matching in regions of epipolar tangency is prevented, this geometrical situation can be used in the reconstruction in other ways. When a projection of a 3D curve in one camera is tangential to an epipolar line, its projection on the other camera must also be tangential to the epipolar line in that camera (Cipolla and Giblin 2000). This fact can be used in the matching algorithm or to correct matching errors as subsequently described.

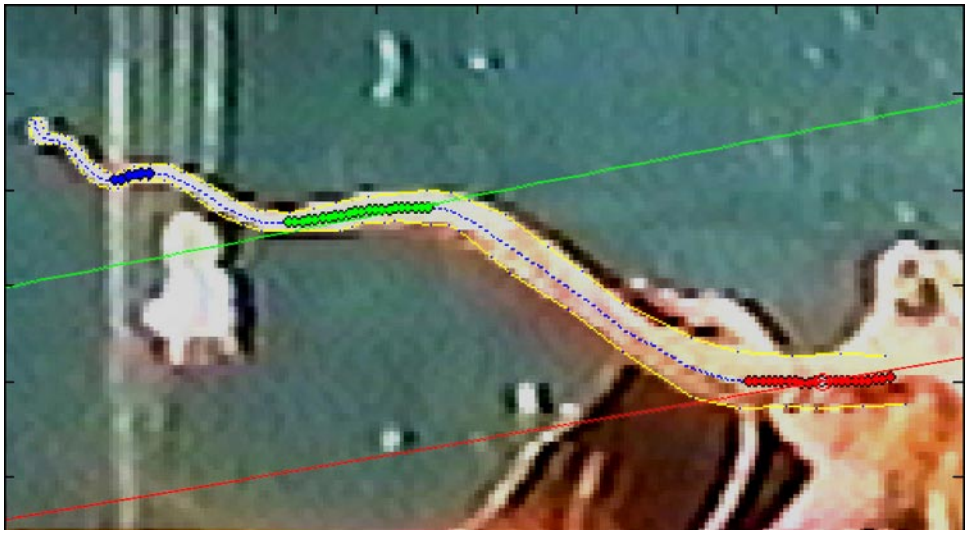

FIG. 6. An octopus arm marked with lines to show areas of epipolar tangency. Blue, green, and red regions mark areas of the midline curve that form a very low angle to the self-epipolar line (see text for explanation). Large zoom used here (×4) was necessary due to the small size of the arm in the visual field. Poor image quality does not degrade the accuracy of the analysis.

The matching algorithm

The algorithm we use to match the two midline curves has two stages: *A*) finding the first matching point and *B*) finding all other matches. We use the following assumptions and preliminaries:

- Midline curves are sampled at equal distances to form sets of equidistant points. Thus the exact location of intersection between a line and such a curve can be between the curve points.
- The number of points for the left and the right curves is generally not the same.
- In part A of the matching algorithm, when a line intersects a curve in more than one place, we take the intersection nearest to the tip (in terms of distance along the curve). Because we define the tip index as 1, the nearest intersection is the one with the lowest index.

The algorithm is composed of the following steps: *A*) Finding the first match, based on the four different cases as shown in Fig. 7 and *B*) Matching the two curves: Start with the match found in *A* and calculate its 3D point in space. For every point on the left midline:

1) Calculate its epipolar line.

2) Find the intersection between the epipolar line and the right curve.

3) If there is more than one such intersection, find the best candidate using two stages. *a*) Calculate the distance along the right curve (arc length distance) between the previous matched point and all candidates. *b*) Reconstruct the 3D points for all candidates and choose the one at smallest distance from the previously reconstructed point (for the first time it would be the known match from *A*). Use this point only if the 2D distance found for it is below some user defined threshold; otherwise discard it. Return to stage 1 for the next point on the left curve, until all points are matched. The algorithm runs over the points of the left curve (the leading curve) and finds their matching points on the right curve (the secondary curve). In some cases it is better to switch the curves. For example, if the orientation of some part of the arm in the leading curve is almost parallel to the line of sight in one view, its projection may be too short for the matching algorithm. In our implementation, the user decides which is the leading curve.

Regions of epipolar tangency (defined by the local angle between the curve and its self-epipolar line) are skipped and not reconstructed. The gaps in the reconstruction are filled later by interpolating from nearby nontangential areas.

Errors in the matching process and the reconstruction

So far, we have assumed that the projection matrices and the epipolar geometry are correct (perfect calibration) and that the 2D midlines are correct (i.e., perfect projections of the real 3D midline of the arm), although of course there are errors in realistic situations. There were two main sources of error in our setup. The first type of error occurs when reconstructing points outside the calibrated space where the accuracy of the DLT algorithm is degraded (Chen et al. 1994; Hinrichs and McLean 1995; Kwon 1999; Kwon and Casebolt 2006). Usually we avoid this situation by moving the calibration frame in the aquarium to cover most of the work space and then choosing the specific position for local calibration that best matches the movement to be analyzed. However, when the position of the arms during motion does not fit the calibrated space, the calibration matrices and the epipolar geometry used are erroneous, leading to errors in the matching of the two 2D midlines and in the 3D reconstruction (Fig. 8).

The second source of errors is inaccuracies in the 2D midline curves. Such errors are more likely to occur when image quality is poor and when parts of the arms are occluded.

Both types of errors may result in incorrect matching and distorted reconstruction. This is especially evident in areas of epipolar tangency, where incorrect matching may cause the reconstruction to fail altogether and, even if it succeeds, the resulting 3D curve will be noncontinuous (Fig. 8, *E* and *F*).

Correcting wrong matches

We use the same method to correct all matching errors. Novel matching points from regions of epipolar tangency are found and the projection matrices are recalculated to minimize the discrepancies in the matches:

For each point on the left 2D curve a self-epipolar line is found and the angle between that line and the curve is calculated. If this angle is below some user-defined threshold, it is considered a tangency point (Fig. 6). All adjacent tangency points are grouped together to form areas of tangency. The number of groups should be the same for the left and right curves. The middle of each such group is used as a matching point. For each matching pair ($\mathbf{p_1}$, $\mathbf{p_2}$), we calculate the distance between the epipolar line $\mathbf{l_2}$ and the point $\mathbf{p_2}$ that should lie on this line. The sum of the distances for all pairs is used as the error measure of the current calibration. Because we do not know the 3D

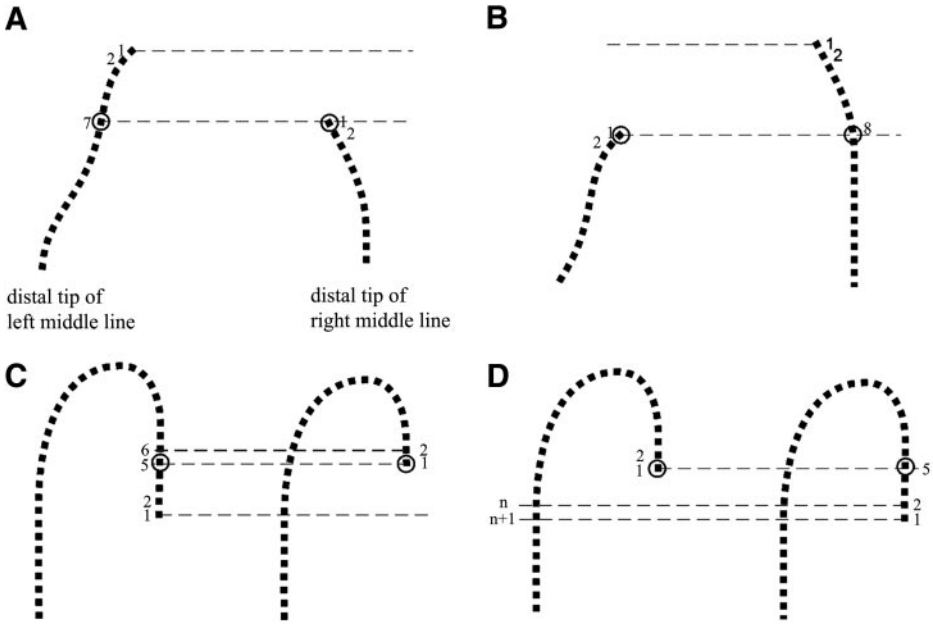

FIG. 7. Finding the first matching points using the epipolar geometry. Distal tips of the left and right midlines are shown in 4 cases. Numbers indicate the indices starting from 1 at the distal tip. Proximal part of the curves is not shown. After finding the correct first matching points (marked by circles), the curves are cropped distally to the match. *A*: there is no intersection for the epipolar line of left point 1 with the right curve, so the intersection of the epipolar line of right point 1 with the left curve is used. *B*: symmetrical case of *A*. *C* and *D*: in both cases there are intersections for both the left and the right epipolar lines. Two cases are distinguished by the index order of the intersections for the epipolar lines of right 1 and 2 points and the left curve. In *C*, the intersection index increases (from 5 to 6) and in *D* the intersection index decreases (from $n + 1$ to n). Note that there can be more than one intersection for each epipolar line with the left curve (with the proximal part of the curve that is not shown). In such cases the first intersection (that nearest to the tip, indexwise) is taken.

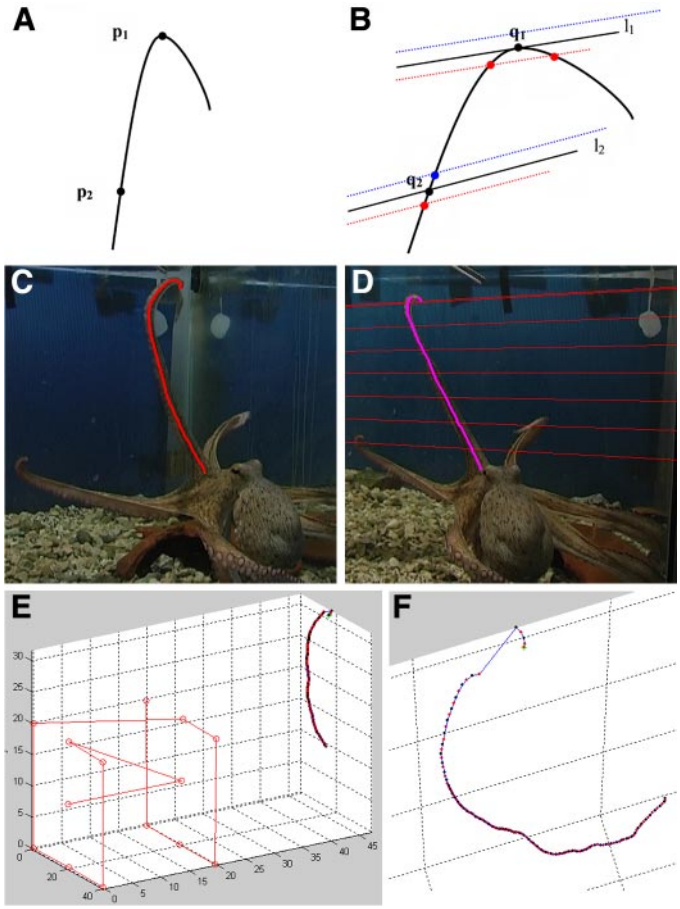

FIG. 8. Epipolar tangency and matching errors. *A*: a part of the left midline curve with two points $\mathbf{p_1}$ and $\mathbf{p_2}$, marked to illustrate the problem. *B*, *right view*: each point has an epipolar line ($\mathbf{l_1}$ and $\mathbf{l_2}$), and a matching point ($\mathbf{q_1}$ and $\mathbf{q_2}$). An error in the position of the curve (especially in the direction normal to the epipolar lines) or an error in the epipolar geometry moves the epipolar lines up or down relative to the curve (blue and red lines). Because $\mathbf{l_1}$ is tangential to the right curve, moving it up will leave $\mathbf{p_1}$ with no matching point, whereas moving it down will result in 2 candidate matches for $\mathbf{p_1}$. Either way the 3D reconstruction will be wrong, with a discontinuity at the area of epipolar tangency. *C–F*: example of the problem of epipolar inaccuracy leading to matching error and discontinuity in the reconstruction. *C*: midline of the arm in the *left view* (red dots). *D*, *right view*: a subset of the epipolar lines (red lines). Intersection points between the epipolar lines and the midline of the arm (magenta dots). Note the small gap near the tip of the arm. *E*: 3D reconstruction of the arm and the calibration frame. Arm was outside the calibrated space. *F*: 3D reconstruction rotated and enlarged to show the discontinuity at the area of epipolar tangency, near the tip.

points (whose projections are the matching points), we cannot follow the simple calibration technique described earlier. Instead, we use a nonlinear optimization technique (Matlab's *fminsearch* function) with our current calibration matrices as an initial guess, to find the two calibration matrices that minimize the matching distance.

Reconstructing arm torsion

Until now we have dealt with the problem of constructing a 3D description of the arm using the midline representation. However, additional information must be incorporated to uniquely describe the arm motion. This is the change in orientation of the arm along its long axis, termed here arm torsion, which may be readily apparent from the direction of the suckers (Fig. 2). We use this information by marking the 2D direction of the suckers relative to the midline of the arm in the two views (Fig. 9). The final results are then displayed as a 3D strip attached to the midline (Fig. 3, step 5).

Automating the process of sucker detection and direction reconstruction seems a very ambitious goal. Even for an expert human operator it is a difficult task. It involves knowledge of the physical structure of the arm to overcome occlusions and uncertainties and enforce continuity.

Constructing a coherent description of motion: registration of consecutive curves

After the successful application of previous stages for every frame of a movement sequence, we perform one final step of aligning the consecutive curves. This is usually needed because there is a problem of uncertainty about the position of the reconstructed curve along the real arm. We are unable to start marking an arm at the same point (the same physical location on the arm) on every frame for the whole sequence. This is because usually there is no one such distinct point near the base of the arm that is visible in all video frames. Even if there is a clear mark on the arm, it is sometimes occluded by the arm itself or by other arms or parts of the octopus' body during the movement.

The solution we chose was to register (find correspondences and align) the 3D reconstructed curves from frame to frame based on the appearance of the arms. For each frame we reconstructed a 3D backbone curve with the maximal length possible. Then we projected back the 3D curve on the left and right images to sample the texture maps along the two 2D backbone curves. We sampled texture by creating a curved 2D coordinate system aligned to the arm. This coordinate system constitutes the backbone curve, as the major axis, and lines that are orthogonal to it (Fig. 10 *A*). The sampled texture map is a rectified 2D description of the light pattern reflected by the octopus arms onto our cameras (Fig. 10*B*). Assuming that small changes in the position from frame to frame do not change the textural information very much, it is then possible to find the best match between two consecutive texture maps using translation along the main axis (the backbone of the arm). This yields a translation value (where the correlation between the maps was maximal) that serves to compensate for the difference in the initial reconstruction point along the arm between two consecutive frames. The same process is re-

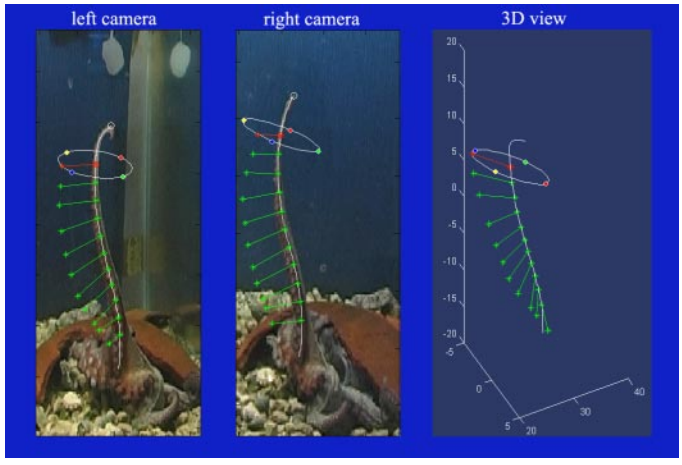

FIG. 9. Snapshot of the process used to reconstruct the direction of the suckers along the arm. A 3D view of the midline is shown on the *right* (white curve). Green lines pointing outward from the midline were already marked by the user. These lines represent the reconstructed direction of the suckers along the midline. White circle is a cursor that can be moved along the midline and its direction is always perpendicular to the local tangent vector of the midline. A line cursor (red line) is directed from the center of the circle outward and can be rotated around the midline. Four colored dots have a fixed position on the circle and are used as landmarks helping the user in orienting the red line in all views. All graphical elements are projected onto the 2 camera views (*middle* and *left*) simultaneously.

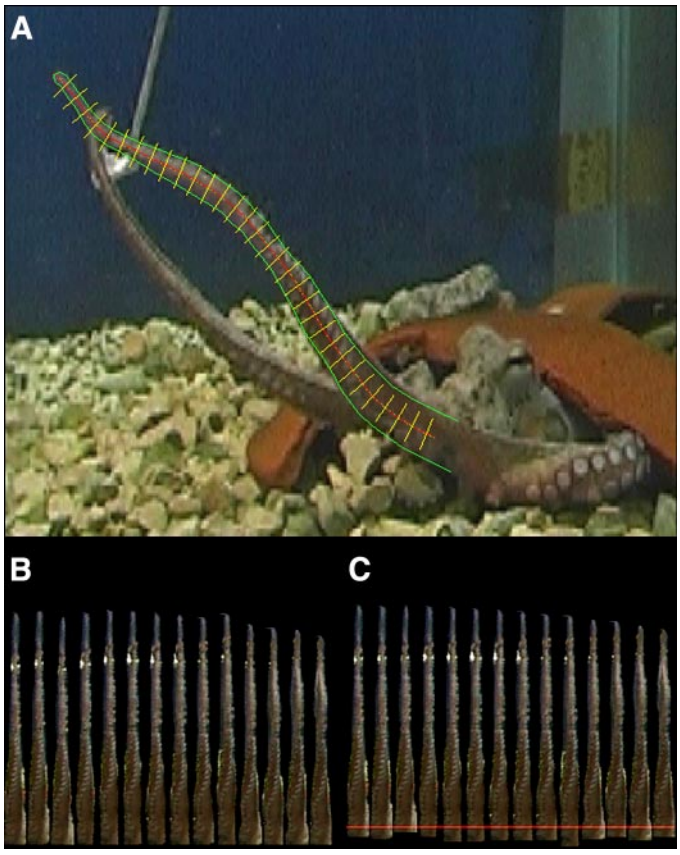

FIG. 10. Finding correspondences between consecutive images of the arms. *A*: octopus in one view (×2 zoom factor). Superimposed on the arm is a curved coordinate system based on the midline (red), lines orthogonal to it (yellow), and the manually marked contour (green). *B*: texture maps of the arm as sampled using the curved coordinate system for 14 consecutive time frames. *C*: texture maps aligned (by translating up or down) using the highest correlation values between consecutive texture maps. Red line marks the most proximal location along the arm that is seen in all frames.

peated for every pair of consecutive frames until the end of the sequence. It is then possible to find the most proximal common point for the whole set of reconstructed curves for each camera (Fig. 10*C*). We clip all curves that have parts more proximal to that point, losing that information, to obtain a set of curves that start at the same physical point along the arm. This process is carried out for each view independently and then the two views are registered and the clipping can be done for the 3D curves. The result is a set of 3D curves that start at the same point on the octopus arm, ready for further analysis.

RESULTS

First, we present an estimation of the reconstruction error in our system. Then, using a simulation of the optics we estimate the contribution of light refraction to our overall error. Last, we present several reconstructions of octopus arm movements.

Estimation of reconstruction accuracy

RECONSTRUCTION OF A TILTED CALIBRATION FRAME. We filmed the same object used for calibration in different orientations, tilted relative to the original orientation. The LEDs of the original orientation were marked and used for calibration and those of the tilted orientation were marked and used for 3D reconstruction. To calculate the reconstruction error (the distance between the reconstructed points and their known 3D positions) we first had to align the reconstructed points of the tilted orientation with the points of the original orientation. The alignment was done in two steps. First, using principal component analysis (PCA), for each of the orientations (original and tilted) we found the three orthogonal axes that best account for the variance of the data. We then took the coordinates of each of the orientations (original and tilted) along its PC axes. To find the best alignment, we used a simple iterative search for the rotations that minimize the sum of 3D distances between the PC coordinates of the two orientations. Note that all manipulations used for the alignment did not change the shape of the reconstructed object but only rotated it. After alignment we calculated the average reconstruction error per point, which lay between 0.3 and 0.4 mm, approximately 1.5-fold larger than the reconstruction error calculated for the original orientations, i.e., for the same set of points used for the calibration.

RECONSTRUCTION OF AN ELONGATED OBJECT. We used a standard 30-cm, white plastic ruler, painted with black rectangular ticks (size 2 × 4 mm, 1 cm apart) along one of its edges. We took images of this ruler using the same setup as for all other experiments reported here. The ruler was situated in different locations and orientations relative to the calibrated space (Fig. 11). Manually digitizing the image of these small rectangles (sometimes as narrow as 2 pixels) proved to be much more difficult and less accurate than either marking the LEDs of the calibration frame or the contour of the octopus arm. The LEDs are circular and are much brighter, enabling easier marking,

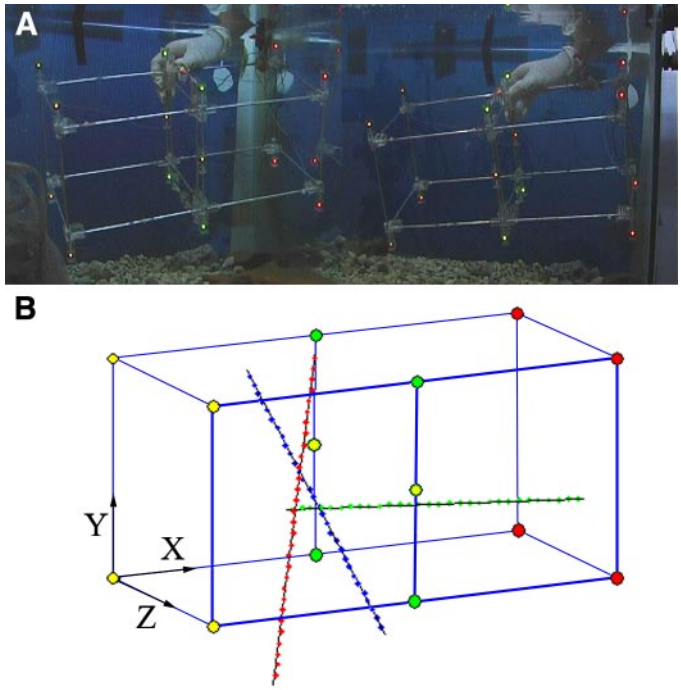

FIG. 11. *A*: *left* and *right views* of the 40 × 20 × 20-cm calibration frame with the light-emitting diodes (LEDs) turned on. Two cameras view the same aquarium facet with an angle of about 40° between the principal rays of the cameras. *B*: 3D reconstruction of a 30-cm ruler in 3 different orientations: X-dominant (green dots), Y-dominant (red dots), and Z-dominant (blue dots). Z-dominant orientation is about 45° relative to the Y-axis in the Y–Z plane. For each orientation, its best-fit 3D line is shown in black. Reconstructed points are superimposed on the 40 × 20 × 20-cm calibration frame (LEDs shown as colored circles connected by blue lines; nearer facet shown by thick lines).

even to subpixel accuracy. The contour of the octopus arm has a smooth and continuous shape, making the decision where exactly to mark much easier.

We used two measures to analyze the reconstruction accuracy: the distance between adjacent points (Table 1) and the distance of the points to their best-fit line (Table 2). Marking and reconstructing a line that is parallel to the Z direction, from the cameras inward, almost parallel to the line of sight, is not possible. We therefore used a tilted orientation of about 45° with respect to the direction of gravity and termed it the Z-dominant direction. Even for this orientation, the ticks were hard to distinguish and those that were far from the cameras were very small (<2 pixels wide). As a result the error in marking their position was largest. The two other directions were easier to detect and therefore more accurate. For a reference we also present here the errors after spline smoothing of the 2D data.

The two types of errors (in length measurements and deviation from linearity) depended differently on the orientation of the ruler. Measuring the length between points was most accurate in the Y direction, average in the X direction, and worst in the Z direction. Deviation from linearity was smaller for the Z and Y directions and larger for the X direction. Smoothing the data improved the accuracy for the Z and Y directions more than for the X direction.

Light refraction simulation

Here we simulate the optical system to estimate the contribution of light refraction to the reconstruction error in our setup.

The simulation uses the same pinhole camera model described earlier and the following parameters: positions of the optical center of the two cameras; the distance between the optical center and sensor plane of each camera; the index of refraction for air, glass, and water; the position of the aquarium facet; the thickness of the glass; and the position of the points to be reconstructed inside the aquarium (Fig. 12). The path of light from a 3D point inside the aquarium to the camera plane is constructed of three straight line segments—one for each medium (Fig. 12). The relation between the indices of refractions and the ratio of angles between these segments is given by Snell's law, and using it gives the exact intersection points of the light rays with the inner and outer sides of the glass. Then, the apparent position of a 3D point is given by continuing the direction of the two rays from camera centers to the aquarium outer surface and finding the intersection of these lines (Fig. 12, **q**′). Note that the distance between the original 3D point and the apparent 3D point is not the reconstruction error, which is much smaller, because the calibration step compensates for some of the differences (mostly canceling translational effects), as subsequently demonstrated.

TABLE 1. *Error in length measurements*

	Dominant Direction		
	Z	Y	X
Original data			
RMS error	9.7	3	3.3
Max error	20.4	7	9.5
Smoothed data			
RMS error	2.4	1.7	1.8
Max error	5	3.2	4.8

Values are presented as percentage of the actual size. Because the ticks were 1 cm apart, a 10% error is a 1-mm deviation.

TABLE 2. *Distance of the reconstructed points from a straight line*

	Dominant Direction		
	Z	Y	X
Original data			
RMS distance	0.09	0.11	0.17
Max distance	0.15	0.29	0.36
Smoothed data			
RMS distance	0.05	0.06	0.16
Max distance	0.1	0.12	0.33

Values are in centimeters.

We placed the coordinates of our calibration frame in the simulated aquarium and calculated the projection of each of the 3D points onto the two camera surfaces, taking into account the refraction of light. We then performed calibration; i.e., using the two sets of 2D points and the known 3D coordinates of the calibration frame we found the projection matrices for each camera. Simulating our experimental setup (60 cm between cameras; 90 cm between camera centers and the aquarium; 0.5 cm width of the glass; 1.0, 1.33, and 1.52 indices of refraction for air, water, and glass, respectively; calibration frame in front of the cameras, positioned 50 cm into the aquarium) we achieved a reconstruction error of about 0.05 cm (averaged error per point, for the calibration points themselves). This is four- to sixfold smaller than the typical results we obtained for physical calibration.

Next we checked how the reconstruction error varies with the distance from the calibration frame. We projected a dense grid of 3D points onto the two camera surfaces and reconstructed their positions. The distance between each recon-

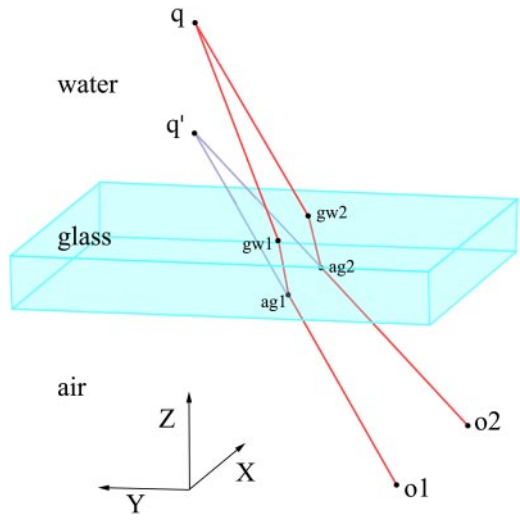

FIG. 12. Simulation of the light refraction. A 3D point **q**, situated inside an aquarium filled with sea water, is projected to the 2 cameras through their optical centers (**o1**, **o2**). Due to light refraction at the interface between water and glass, and between glass and air, the actual path of light is composed of 3 line segments. Continuing the direction of the segments **o1**–**ag1** and **o2**–**ag2**, and finding their intersection gives **q**′, the apparent position of the point. Here we changed the parameters of the simulation to extreme values just for display purposes (glass width 32 cm instead of 0.5 cm, glass index of refraction 2.50 instead of 1.52 and cameras at the side of the aquarium instead of in front of the center of the aquarium).

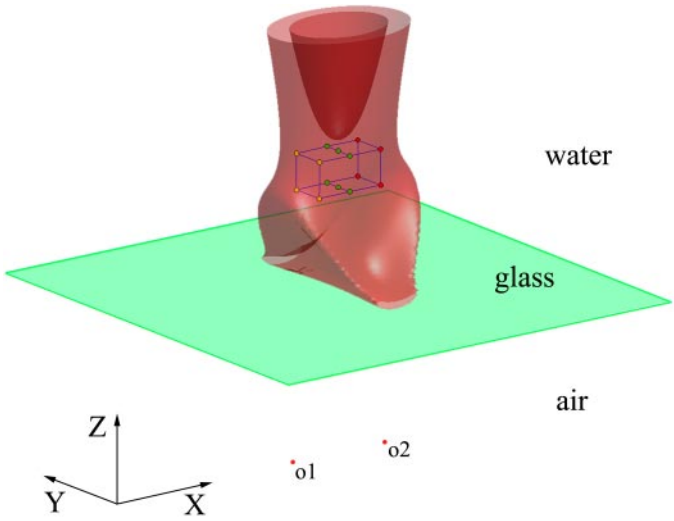

FIG. 13. Simulation result: the error surface of 2-mm reconstruction error. Each point of a dense grid around the calibration frame was projected onto the 2 cameras, taking into account the effect of light refraction. 2D matching points were reconstructed, and the distance of each reconstructed point from the original 3D point was calculated. Error surface has a complex shape that depends on the geometry of the experimental setup. Simulation results show that the contribution of light refraction to the overall reconstruction error is small. Points that are inside a rectangular space almost twice the size of the calibration frame have reconstruction errors of <2 mm.

structed point and its known 3D source was calculated as the local reconstruction error, and an isosurface of 0.2 cm error was presented (Fig. 13). The shape of this closed surface depends on the relative position of the cameras, the calibration frame, and the aquarium facet. Although the shape itself is complex, it supports the statement that as long as we reconstruct objects inside the calibrated space or in its immediate proximity, the errors arising from light refraction are small relative to our overall error.

Last, we simulated the ruler experiment (described earlier), to estimate the errors in length measurements. In the simulation we positioned a 3D cross centered at the middle of the calibration frame. The 3D cross was composed of three straight lines, each parallel to one of the axes. Each line was composed of 60 evenly spaced points, 1 cm apart. The 3D cross was reconstructed and distances between neighboring points along the three major axes were calculated. These distances were compared with the given value of 1 cm. The error in length measurements varied with orientation and position (Fig. 14 *A*). For points inside the calibrated space, the error was <0.55% for the Z axis, <0.02% for the Y axis, and <0.2% for the X axis.

The reconstruction error also varied with orientation and position (Fig. 14*B*). For the Z axis, the error was <1.7 mm for points inside the calibrated space. For the Y axis it was <1.1 mm for points inside the calibrated space, and for the X axis it was <1.1 mm for the whole range.

CAN WE USE EPIPOLAR GEOMETRY? We used the simulated environment to check the influence of light refraction on the accuracy of epipolar lines, using two independent ways of doing the calculations. First, by calculating the epipolar line, using the fundamental matrix, as done in our experiments and, second, by simulating the projection of points composing the epipolar line, taking light refraction into account. This was done for a grid of 27 locations, from a rectangular box twice the size of the calibration frame. For each location we calculated the distance between the calculated epipolar line and the projected points. The maximal error was 0.6 pixel and average error 0.2 pixel. Both errors are small, validating the use of epipolar geometry in our system.

Reconstructions of octopus arm movements

1) Reaching movements (Fig. 15, A–D). In both examples shown here, the backbone curve of the arm is almost always confined to one plane. The curve is planar in most of the frames and the plane stays the same during the movement. The plane of movement includes the line that connects the octopus eyes and the target. Usually, it is also aligned with the direction of gravity. Note, however, that the initial stages of the reaching movement may have a phase where they lie outside this plane. This occurs when the creation of the bend is accompanied by a twist movement in the arm. The bend then travels along the arm, straightening it as it moves. The distal part of the arm appears to be passively pulled along by the bend. The position of the bend is clearly seen in the reconstructions. Note, however, that in Fig. 15*C*, another bend is formed near the base of

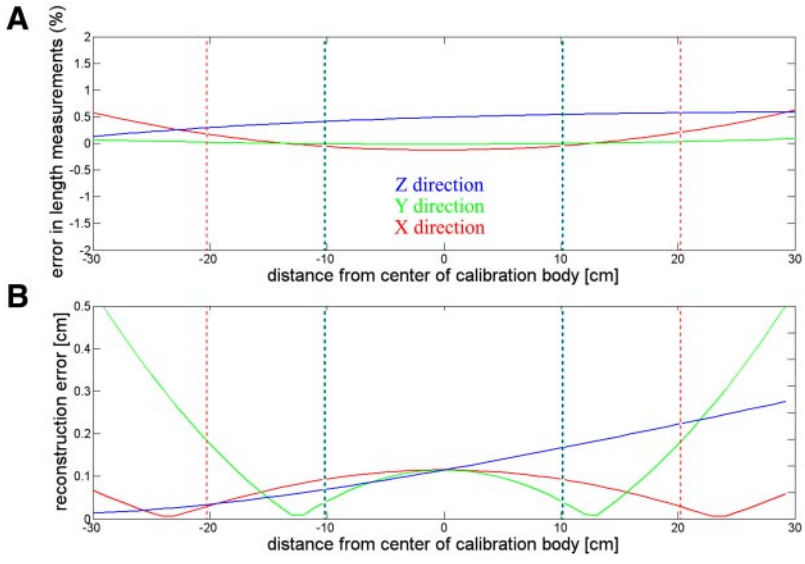

FIG. 14. Simulation result: errors in reconstructing straight, 60-cm lines. Three lines were oriented along the 3 major axes and centered at the middle of the calibration frame. Red, green, and blue show the X, Y, and Z directions, respectively. Dashed vertical lines show the size of the calibration frame in each direction. *A*: error in length measurements (%) as a function of the distance from the center of the calibration frame. *B*: reconstruction error (cm) as a function of the distance from the center of the calibration frame.

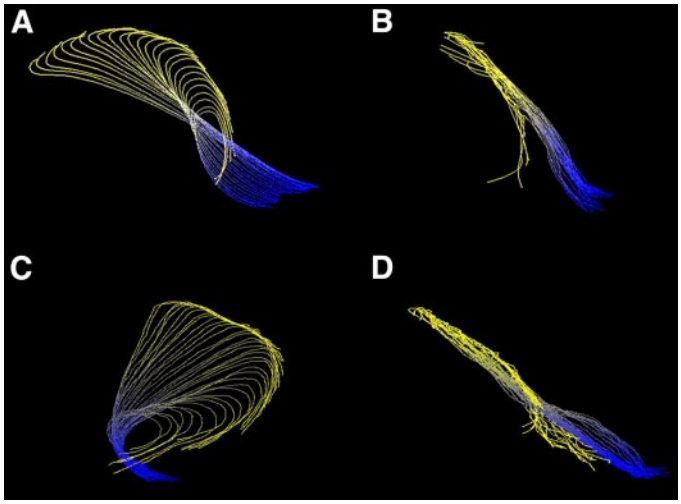

FIG. 15. *A*: reconstruction of 0.5 s of a typical reaching movement represented by about 25 3D midline curves. Distance along the arm is color coded from blue (arm base) to yellow (arm tip). Initial shape of the arm is highly curved. During the movement the bend propagates toward the tip and the arm is straightened. When the bend reaches the tip (about the 10th curve from the last of this example), the arm is fully stretched. *B*: reconstruction in *A* is rotated approximately 90° around the axis of the fully stretched arm to show that the motion is planar. *C* and *D*, as in *A* and *B* for another reaching movement.

the arm at the late stages of the reaching movement. This bend does not travel along the arm and it is similar in appearance to a joint in an articulated arm.

2) Initiation movement. When a bend is created in the octopus arm it is usually accompanied by a noticeable twist (torsion movement) along the arm (Fig. 16). Such a twist usually breaks the planarity of the arm (evident as a local helix of the arm) as well as pointing the suckers toward a possible target (a twist of the suckers around the midline) (Mitelman et al. 2005). Bend initiation occurs not only before reaching movements. The octopus in Fig. 17 created a bend while starting to move to the left. Note that in this example the bend is created without rotating the suckers outward.

DISCUSSION

This article describes a system for 3D reconstruction of octopus arms during natural motion. Its successful application to several types of octopus arm movements extends our ability to study novel aspects of these movements. In addition, the tools we developed for 3D motion reconstruction can be applied to any elongated, line-symmetric, soft object.

We envision using the system for a detailed study of complex arm movements, such as manipulation of objects, and for reconstructing a large number of movements to create a database of octopus movements. We believe that this 3D reconstruction system, together with the advanced electrophysiological and modeling techniques that are already available, will make a major contribution to the research of movement control of muscular hydrostats.

The main bottleneck in reconstructing a large number of movements is the time-consuming task of manually marking arm contours. One means of saving time and increasing the robustness of the 3D reconstruction is to use *model-based* tracking and reconstruction. A nice example of this approach is the system developed by MacIver and Nelson (2000) to track movement and posture of an electric fish. They used a 3D polygonal model of the fish based on a graphical user interface that allows the user to translate, rotate, and deform the model to fit it to the digitized video images of the animal. For a much simpler object—the whisker of a rat—efficient, semiautomated tracking was demonstrated by Knutsen et al. (2005). Their system is capable of accurate 2D tracking of head and whisker movements in freely moving rodents using high-speed video. The shape of the octopus arm varies much more than the whisker of a rat or the body of the electric fish, requiring other, more sophisticated techniques for speeding up tracking and reconstruction.

In some behavioral setups it is possible to control for the contrast between the animal and the background so that segmenting the arm from the background and tracking its movements becomes easy, as in the system described by Baek et al. (2002) for the analysis and classification of nematode behavior. However, for the setup reported here, with two cameras in stereo configuration, the possibility of controlling the back-

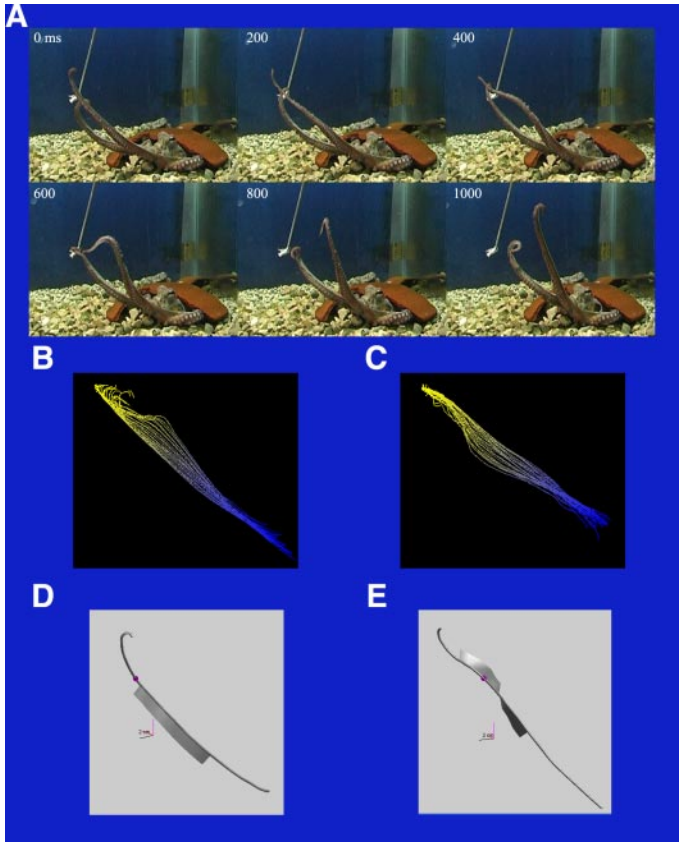

FIG. 16. *A*: 6 snapshots of a bend initiation movement. At 0 ms the upper arm is straight and the suckers point downward. Later, a twist is evident (200 ms, about halfway along the arm) and the suckers rotate and point upward (400 ms). A bend is apparent (600 ms), the suckers point outward as the bend propagates along the arm (800–1,000 ms). *B*: 3D reconstruction of the first 500 ms of the movement represented by 25 midline curves. Point of view is similar to that of the images in *A*. *C*: 3D reconstruction is rotated 90° around the axis of the straight arm of 0 ms. Fold that is created in the arm before the propagating bend is evident both in *B* and in *C*, indicating that the arm assumes a local helix structure. *D*: 3D reconstruction of the arm with the direction of the suckers (shown as a tape attached to the midline) at *time 0* ms. *E*: like *D* but at time 400 ms. Note the change in the direction of the suckers from pointing almost down to pointing upward over a length of less than a third of the arm. In this bend initiation movement the local helix structure (*B* and *C*) is accompanied by a twist around the longitudinal axis of the arm (from *D* to *E*).

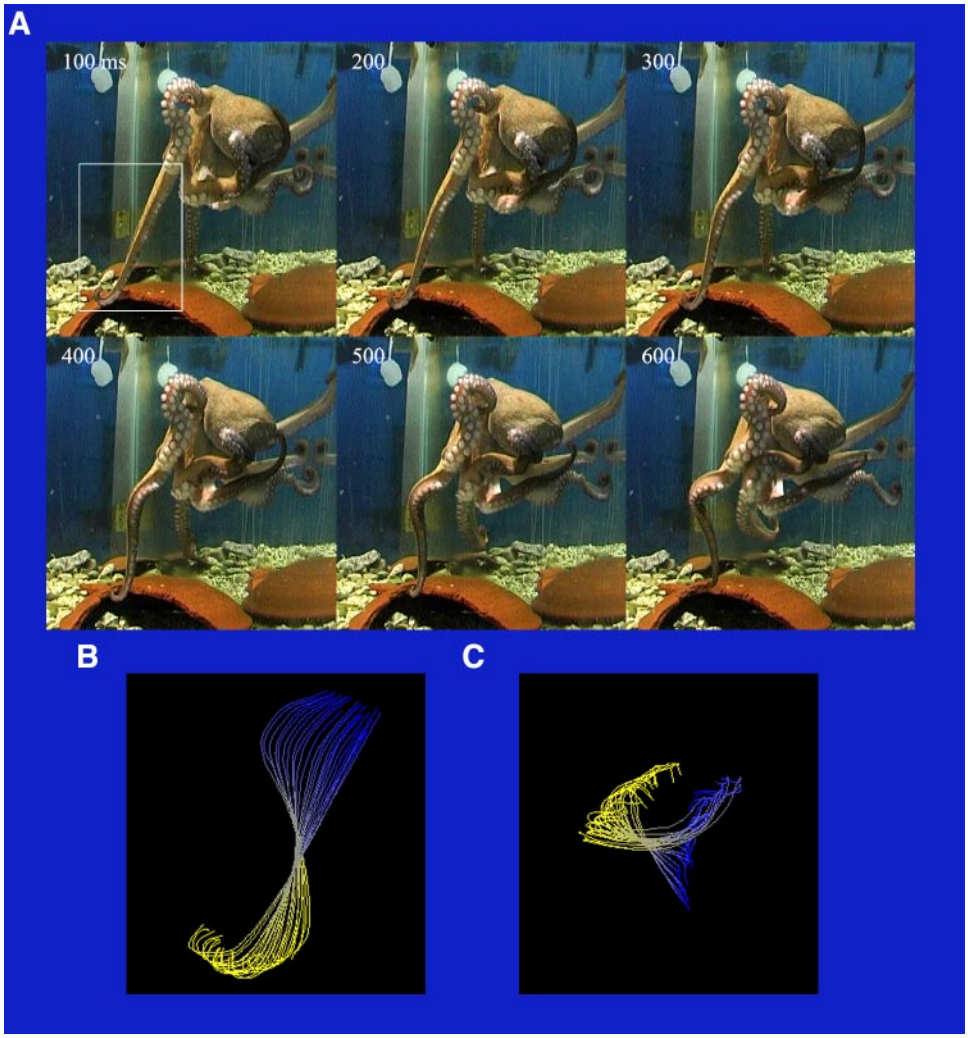

FIG. 17. *A*: 6 snapshots of a bend initiation movement. At 100 ms the arm, marked by a white rectangle, is straight. Later, a bend is formed in the arm (200 ms, about halfway along the arm). Octopus moves slowly to the left and the bend gradually becomes more significant (300–600 ms). *B*: 3D reconstruction of the first 600 ms of the movement represented by 30 midline curves. Point of view is similar to that of the images in *A*. *C*: 3D reconstruction is rotated to show the arm from the direction of the tip toward the base.

ground is rather limited, making automatic segmentation and tracking a bigger challenge.

We recently applied the approach of texture segmentation by multiscale aggregation (Galun et al. 2003, 2005) to segmenting octopus arms from video data (Zelman et al., unpublished observations). This seems to us the most promising way toward an automatic version of midline extraction and a full arm reconstruction system.

The reconstruction of a 3D curve from its 2D projections here utilized a bottom-up approach, using epipolar geometry to find the best match between the 2D projections. It is a straightforward and accurate method but with some inherent problems of uncertainty in areas of epipolar tangency. Other approaches can be used to solve these problems. de Groot et al. (2002, 2004) estimated the longitudinal axis of a snake's tongue by fitting a polynomial approximation of a 3D curve to the 2D projections. This essentially circumvents the need for matching between the 2D projections and enforces smoothness and continuity on the result. However, this method may suffer from problems of optimization techniques, such as getting caught in a local minima (thus giving the wrong solution), especially when attempting reconstruction of complex shapes. It may prove beneficial to combine both approaches to achieve a robust and accurate method for 3D reconstruction of line-symmetrical soft bodies.

Videotaping animal behavior in an aquarium raises the problem of light refraction. Here we did not present a general solution for the problem but rather used methods that minimize refraction errors. We presented experimental evidence for the validity of our method and a simulation study that estimates the overall contribution of light refraction as less than half of our reconstruction errors.

Another factor influencing the accuracy and robustness of the reconstruction is the number of cameras used. Using more than two cameras may solve most of the problems of occlusion and epipolar tangency. Nevertheless we used two cameras because they were easier and cheaper to synchronize, the physical setup was simpler, and the amount of storage needed for video data was lower. As technology progresses and prices drop it will be possible to improve the system by using at least three cameras.

As for the reconstructions presented here, we were able for the first time to study the changes in the shape of the octopus arm during motion. The kinematic studies of Gutfreund et al. (1996, 1998) and Sumbre et al. (2001) showed that, during reaching movements, the path of the bend point is not straight but curved. This allowed for the calculation of a best-fit plane for the path. In most cases the deviations from that plane were small, showing that the common path is almost planar. The 3D reconstructions presented here showed that the midlines of the

arm themselves move within a plane for most of the duration of the movement. There may be a phase of movements outside this plane at the beginning of the movement, when the distal part of the arm is curved to the right or left of the plane of motion. It is probable that the tip is pulled by the bend and slides under the influence of water drag into the plane of motion.

We were also able to reconstruct bend initiation movements and observed two different categories: bend initiation with a twist along the arm (torsion movement) that precedes a reaching movement and bend initiation that does not include any torsion component. The latter is obviously comparable to the creation of the pseudojoints during the fetching movement (Sumbre et al. 2005; and Fig. 1*B*).

What muscles are activated to execute the different types of bend initiation movements? The anatomy of the arm suggests that torsion movements involve activation of helical muscles. However, such activation can give rise to many different movements depending on the combination of activations of other muscles. A straight arm can twist while remaining straight or it can change its shape into a helix curved around its previously straight shape. Because measuring the activity of different muscle groups during movement is not yet possible we may resort to modeling. Testing hypotheses about muscle activation and arm dynamics should naturally be carried out using a 3D computer model that can simulate arm biomechanics. Our current 2D dynamic model (Yekutieli et al. 2005) is insufficient for these purposes; therefore a 3D model that incorporates helical muscles and is general enough to simulate muscular hydrostat movements is currently being developed (Karra 2006).

APPENDIX A

Finding the midline (medial axis) of the arm using the grass-fire algorithm

Our implementation uses a discrete approximation to wave propagation and collision. Two waves are initiated from each side of the contour and the locus of their collision is the midline of the contour. The input to the algorithm is the set of points representing the contour, from the base to the tip and back along the other side to the base. The algorithm has the following steps.

1) The contour and the propagating waves are represented as integer values in a 2D matrix. To achieve the desired accuracy and smoothness of the midline the contour is up-sampled so its bounding box will have ≥25,000 cells in the 2D matrix (the matrix can be viewed as an image, initially having only color 0 pixels).

2) The contour points are inserted into the 2D matrix and the gaps between the points are closed by line segments, creating a continuous curve. Two distinct values (color 1 and color 2) are used for the two sides of the contour. The two sides of the contour are connected at the base using color 3 pixels. The result is a closed curve.

3) The inside of the closed shape is filled with the color 4, to create a distinction between inside and outside the shape (color 0).

4) From each side of the contour, a wave is propagated toward the inside of the shape: all inside pixels (color 3) that have neighbors with color 1 or 2 change their color to that of their neighbor. This step is repeated until there are no more color 3 cells. The result is the shape of the arm, with one half color 1 and the other color 2. The border between these two colors is the midline of the shape.

5) The border is detected: all color 1 pixels that have color 2 neighbors change their value to a new value (color 5).

6) The color 5 pixels are turned into an ordered list representing the midline. This is done by tracing the color 5 pixels from the tip of the shape, back toward the base.

7) The result of step 6 is smoothed and undersampled using a cubic smoothing spline (Matlab's *csaps* function). The output is a set of almost evenly spaced points. Very small arms (e.g., 20 pixels length) will have a midline with >100 points that are approximately 0.2 pixels apart, whereas very large arms (600 pixels) will have about 300 points that are approximately 2 pixels apart. This way even such extreme cases can be handled by the later stages of matching.

The Matlab implementation of this algorithm uses array operations. Thus in step 4 for example, propagating the color of the two sides of the contour is done by the following code. The original matrix is termed img and its dimensions are sx × sy:

```
FINISHED = false;

while ~FINISHED

  im3 = img==3; % find all color 3 pixels (inside pixels)

  if all (all (im3==0))
    FINISHED = true;
  else %

    im1 = img==1; % find all color 1 pixels

    % shifted matrices to count the neighbors
    im1u = [im1(2:end,:) ; zeros(1,sx) ];
    im1d = [zeros(1,sx) ; im1(1:end-1,:)];
    im1r = [zeros(sy,1) im1(:,1:end-1)];
    im1( = [im1(:,2:end) zeros(sy,1) ];

    % count how many color 1 neighbors each pixel has
    num_1 = im1u + im1d + im1r + im11;

    % paint color 3 pixels with color 1 neighbors to color 1
    img (im3 & (num_1 > 0)) =1;

    % find again all color 3 pixels
```

```
    im3 = img==3;

    if all (all (im3==0)) % are all inside pixels painted
       FINISHED = true; % yes

    else % no, there are some more inside pixels

       im2 = img==2; % find all color 2 pixels

       % shifted matrices to count the neighbors
       im2u = [im2(2:end,:) ; zeros(1,sx) ];
       im2d = [zeros(1,sx) ; im2(1:end-1,:)];
       im2r = [zeros(sy,1) im2(:,1:end-1)];
       im2l = [im2(:,2:end) zeros(sy,1) ];

       % count how many color 2 neighbors each pixel has
       num_2 = im2u + im2d + im2r + im2l;

       % paint color 3 pixels with color 2 neighbors to color 2
       img (im3 & (num_2 > 0)) = 2;

     end
   end
end % of while ~FINISHED
```

The Matlab implementation is very simple, easy to use, and easy to change, but has a much slower runtime (seconds for each image) compared with our older C++ version (~0.1 s for an image). The C++ code is much faster for two reasons. First, C++ complied code is usually faster than Matlab JIT (Just In Time) acceleration and, second, we used a more efficient implementation. Instead of checking the whole matrix in every iteration, a queue was used to process only the neighbors of the desired pixels (a breadth first search). The queue implementation is of course possible to code in Matlab but it proved to be much slower than the version presented here.

The midline extraction algorithm as presented here is not part of the Matlab image processing toolbox. However, some of the morphological operators found in that package can achieve similar results. The Euclidean distance transform finds the distance of each pixel in a binary image to its nearest boundary point (Matlab's *bwdist* function). The ultimate erosion of a binary image uses the distance transform. It is the regional maxima of the Euclidean distance transform of the complement of an image (Matlab's *bwulterode* function). For many simple shapes, the results of ultimate erosion are very similar to the results of our method. It is also about 10 times faster than our Matlab implementation. However, it is not as robust and, in extreme cases, it may fail altogether. The robustness of our approach is achieved by the additional information used by our algorithm: the base of the arm (represented by two points) and the arm tip (one point) to explicitly divide the contour into two sides and propagate the waves from these two sides inward.

Further analysis and an implementation of the grass-fire algorithm using active contours can be found in Leymarie and Levin (1992).

ACKNOWLEDGMENTS

We thank Dr. Jenny Kien for suggestions and editorial assistance and S. Hanassi, M. Moris, N. Bar-Am, Dr. German Sumbre, and Dr. Yoram Gutfreund for advice and assistance.

Present address of Y. Yekutieli: The Faculty of Mathematics and Computer Science, Weizmann Institute of Science, POB 26, Rehovot 76100, Israel.

GRANTS

This work was supported by Defense Advanced Research Projects Agency Grant N66001-03-R-8043, Israel Science Foundation Grant 580/02, and the Moross laboratory. T. Flash is an incumbent of the Dr. Hymie Morros professorial chair.

REFERENCES

Abdel-Aziz YI, **Karara HM.** Direct linear transformation from comparator coordinates into object-space coordinates in close range photogrammetry. In: *Proceedings of the ASP/UI Symposium on Close-Range Photogrammetry*. Falls Church, VA: American Society of Photogrammetry, 1971, p. 1–18.

Aggarwal JK, **Cai Q.** Human motion analysis: a review. *Comput Vision Image Understanding* 73: 428–440, 1999.

Baek JH, Cosman P, Feng Z, Silver J, Schafer WR. Using machine vision to analyze and classify Caenorhabditis elegans behavioral phenotypes quantitatively. *J Neurosci Methods* 118: 9–21, 2000.

Blum H. A transformation for extracting new descriptors of shape. In: *Models of the Perception of Speech and Visual Form*. Cambridge, MA: MIT Press, 1967.

Bodenheimer B, Rose C, Rosenthal S, **Pella J.** The process of motion capture: dealing with the data. In: *Proceedings of Computer Animation and Simulation '97* (Eurographics Animation Workshop), edited by Thalmann D, van de Panne M. Vienna: Springer-Verlag, 1997, p. 3–18.

Chen L, Armstrong CW, Raftopoulos DD. An investigation on the accuracy of three-dimensional space reconstruction using the direct linear transformation technique. *J Biomech* 27: 493–500, 1994.

Chirikjian GS, Burdick JW. A modal approach to hyper-redundant manipulator kinematics. *IEEE Trans Robot Automat* 10: 343–354, 1994.

Cipolla R, Giblin PJ. *Visual Motion of Curves and Surfaces.* Cambridge, UK: Cambridge Univ. Press, 2000.

de Groot JH, van der Sluijs I, Snelderwaard PC, van Leeuwen JL. A three-dimensional kinematic analysis of tongue flicking in Python molurus. *J Exp Biol* 207: 827–839, 2004.

de Groot JH, van Leeuwen JL. Estimation of the longitudinal axis of line symmetrical soft bodies by stereophotogrammetry. *J Biomech* 35: 823–827, 2002.

Faugeras O. Three-dimensional computer vision—a geometric viewpoint. In: *Artificial Intelligence*. Cambridge, MA: MIT Press, 1993.

Fiorito G, Planta CV, **Scotto P.** Problem solving ability of Octopus vulgaris Lamarck (Mollusca, Cephalopoda). *Behav Neural Biol* 53: 217–230, 1990.

Galun M, Apartsin A, Basri R. Multiscale segmentation by combining motion and intensity cues. In: *Proceedings of Computer Vision and Pattern Recognition. CVPR* 2005. Piscataway, NJ: IEEE Computer Society, 2005, vol. 1, p. 1

Galun M, Sharon E, Basri R, **Brandt A.** Texture segmentation by multiscale aggregation of filter responses and shape elements. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision*. Piscataway, NJ: IEEE Computer Society, 2003, p. 716–725.

Gavrila DM. *Vision-Based 3-D Tracking of Humans in Action.* (PhD thesis). College Park, MD: Univ. of Maryland, 1996.

Golland P, **Grimson WEL.** Fixed topology skeletons. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE Computer Society, 2000, vol. 1, p. 10–17.

Graziadei P. The nervous system of the arms. In: *The Anatomy of the Nervous System of Octopus vulgaris*, edited by Young JZ. Oxford, UK: Clarendon Press, 1971, p. 45–59.

Gutfreund Y, Flash T, Fiorito G, Hochner B. Patterns of arm muscle activation involved in octopus reaching movements. *J Neurosci* 18: 5976–5987, 1998.

Gutfreund Y, Flash T, Yarom Y, Fiorito G, Segev I, Hochner B. Organization of octopus arm movements: a model system for studying the control of flexible arms. *J Neurosci* 16: 7297–7307, 1996.

Hartley RI. Estimation of relative camera positions for uncalibrated cameras. In: *Proceedings of ECCV'92, Second European Conference on Computer Vision, Santa Margherita Ligure, Italy*, edited by Sandini G. Berlin: Springer-Verlag, 1992, p. 579–587.

Hinrichs RN, McLean SP. NLT and extrapolated DLT: 3-D cinematography alternatives for enlarging the volume of calibration. *J Biomech* 28: 1219–1224, 1995.

Hughes NF, LH Kelly. New techniques for 3-D video tracking of fish swimming movements in still or flowing water. *Can J Fish Aquat Sci* 53: 2473–2483, 1996.

Jennings C. Robust finger tracking with multiple camers. In: *Proceedings of the International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*. Piscataway, NJ: IEEE Computer Society, 1999, p. 152–160.

Karra TN. *3D Dynamic Model of the Octopus Arm* (MSc thesis). Rehovot, Israel: The Weizmann Institute of Science, 2006.

Kier WM. Hydrostatic skeletons and muscular hydrostats. In: *Biomechanics (Structures and Systems): A Practical Approach*, edited by Biewener AA. Oxford, UK: IRL Press, 1992.

Kier WM, Smith KK. Tongues, tentacles and trunks: the biomechanics movement in muscular-hydrostats. *Zool J Linn Soc* 83: 307–324, 1985.

Kier WM, Thompson JT. Muscle arrangement, function and specialization in recent coleoids. *Berliner Paläobiol Abhandl* 3: 141–162, 2003.

Knutsen PM, Derdikman D and Ahissar E. Tracking whisker and head movements in unrestrained behaving rodents. *J Neurophysiol* 93: 2294–2301, 2005.

Kwon YH. A camera calibration algorithm for the underwater motion analysis. In: *Scientific Proceedings of the XVII International Symposium on Biomechanics in Sports*, edited by Sanders RH, Gibson BJ. Perth, Australia: Edith Cowan Univ., 1999, p. 257–260.

Kwon YH. *Motion Analysis: Camera Calibration*. http://www.kwon3d.com/theory/calib.html, 2007. Online.

Kwon YH, Casebolt JB. Effects of light refraction on the accuracy of camera calibration and reconstruction in underwater motion analysis. *Sports Biomech* 5: 315–340, 2006.

Lagarias JC, Reeds JA, Wright MH, Wright PE. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM J Optim* 9: 112–147, 1998.

Leymarie F, Levin MD. Simulating the grassfire transform using an active contour model. *IEEE Trans Pat Anal and Mach Intell* 14: 56–75, 1992.

MacIver MA, Nelson ME. Body modeling and model-based tracking for neuroethology. *J Neurosci Methods* 95: 133–143, 2000.

Mather JA. How do octopuses use their arms? *J Comp Psychol* 112: 306–316, 1998.

Mazzoni A, Garcia-Perez E, Zoccolan D, Graziosi S, Torre V. Quantitative characterization and classification of leech behavior. *J Neurophysiol* 93: 580–593, 2005.

Mitelman R, Yekutieli Y, Flash T, Hochner B. Towards a better understanding of the octopus' motor control: a kinematic description of the bend initiation movement using 3D reconstruction of the entire arm (Abstract). *Israel Soc Neurosci ISFN 14th Annu Meeting* 2005.

Moynihan M. Conservatism of displays and comparable stereotyped patters among cephalopods. In: *Function and Evolution in Behavior*, edited by Baerends G, Beer C, Manning A. Oxford, UK: Clarendon Press, 1975, p. 276–292.

Packard A, Sanders GD. Body patterns of Octopus vulgaris and maturation of the response to disturbance. *Anim Behav* 19: 780–790, 1971.

Roberts LG. Machine perception of three-dimensional solids. In: *Optical and Electrooptical Information Processing*, edited by Tippett J, Berkowitz D, Clapp L, Koester C, Vanderburgh A. Cambridge, MA: MIT Press, 1965, p. 159–197.

Rowell CHF. Activity of interneurons in the arm of Octopus in response to tactile stimulation. *J Exp Biol* 44: 589–605, 1966.

Sumbre G, Fiorito G, Flash T, Hochner B. Motor control of the octopus flexible arm. *Nature* 433: 595–596, 2005.

Sumbre G, Fiorito G, Flash T, Hochner B. Octopuses use a human-like strategy to control precise point-to-point arm movements. *Curr Biol* 16: 767–772, 2006.

Sumbre G, Gutfreund Y, Fiorito G, Flash T, Hochner B. Control of octopus arm extension by a peripheral motor program. *Science* 293: 1845–1848, 2001.

Wells MJ, **Wells J.** The function of the brain of octopus in tactile discrimination. *J Exp Biol* 34: 131–142, 1957.

Xu G, Zhang Z. Epipolar geometry in stereo, motion and object recognition: a unified approach. In: *Motion-Based Recognition* (*Computational Imaging and Vision*), edited by Shah M, Jain R. Berlin: Springer-Verlag, 1996.

Yekutieli Y, Sagiv-Zohar R, Aharonov R, Engel Y, Hochner B, Flash T. Dynamic model of the octopus arm. I. Biomechanics of the octopus reaching movement. *J Neurophysiol* 94: 1443–1458, 2005.

Yekutieli Y, Sagiv-Zohar R, Hochner B, Flash T. Dynamic model of the octopus arm. II. Control of reaching movements. *J Neurophysiol* 94: 1459–1468, 2005.

Yekutieli Y, Sumbre G, Flash T, Hochner B. How to move with no rigid skeleton? The octopus has the answers. *Biologist* (*Lond*) 49: 250–254, 2002.

Zakotnik J, Matheson T, Dürr V. A posture optimization algorithm for model-based motion capture of movement sequences. *J Neurosci Methods* 135: 43–54, 2004.

Zullo L, Sumbre G, Flash T, Hochner B. Characterization of neural activities of the octopus higher motor centers (Abstract). *Israel Soc Neurosci ISFN 14th Annu Meeting* 2005.