Advanced Robotics, Vol. 17, No. 2, pp. 165–178 (2003)

Also available online - www.vsppub.com

Full paper

Online tracking and mimicking of human movements by a humanoid robot

ALEŠ UDE [1,2,*] and CHRISTOPHER G. ATKESON [1,3]

[1] *ATR Human Information Science Laboratories, Department 3, 2-2-2 Hikaridai, Keihanna Science City (Seika-cho, Soraku-gun), Kyoto 619-0288, Japan*
[2] *Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics, Jamova 39, 1000 Ljubljana, Slovenia*
[3] *Carnegie Mellon University, Robotics Institute and HCI Institute, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA*

Received 30 April 2002; accepted 31 May 2002

Abstract—This paper describes a humanoid robot system that can capture and mimic the motion of human body parts in real-time. The underlying vision system is able to automatically detect and track human body parts such as hands and faces in images captured by the robot's eyes. It is based on a probabilistic approach that can detect and track multiple blobs in a 60-Hz stereo image stream on a standard dual processor PC. A random jerk model is employed to approximate the observed human motion and a Kalman filter is used to estimate its parameters (three-dimensional positions, velocities and accelerations). This enables the system to realistically mimic the perceived motion in real-time.

Keywords: Humanoid robotics; real-time vision; mimicking; imitation.

1. INTRODUCTION

Humanoid robots are similar to humans and can — unlike other robots — carry out human-like movements. This makes programming by demonstration or imitation learning particularly interesting for humanoid robots [1, 2]. We are interested in creating a humanoid robot system that can imitate, practise and generalize human behavior using the information provided by the robot's eyes. This requires a continuous, real-time interaction of a robot with a teacher, thus setting high requirements for a robot perception and motor control system. Once motion perception is seen as a continuous process that interacts with the motor control system, the required standards of reliability become much more stringent because

*To whom correspondence should be addressed at ATR Human Information Science Laboratories. E-mail: aude@atr.co.jp

failure of one of the subsystems might cause the entire system to break down. Here we describe how we deal with the perception of human motion and its online conversion into robot motion. The considered issues include (i) the automatic, real-time detection of objects of interest in images captured by cameras in motion, which sets the groundwork for the future development of a visual attention system, (ii) a probabilistic approach to real-time tracking of human body parts, (iii) human motion estimation based on a random jerk model and Kalman filtering, and (iv) generation of humanoid robot motion using data provided by the vision system. Higher-level issues such as generalization of the observed movements and other cognitive issues are not the topic of this paper.

Our robot *DB* is a hydraulic anthropomorphic robots with a relatively complete body. *DB* has 30 d.o.f.: 7 d.o.f. for each arm, 3 d.o.f. for each leg, 2 d.o.f. for each eye, 3 d.o.f. for the head and 3 d.o.f. for the torso. Each eye of the robot's oculomotor system consists of two cameras: a wide-angle (100° view angle horizontally) color camera for peripheral vision and a second narrow-view color camera (24° view angle horizontally) providing foveal vision. This setup mimics the foveated retinal structure of primates. The images from the wide-angle cameras are captured and processed by a dual processor PC running the Windows NT operating system. The extracted data is sent via a serial line to Power PC processors that run the motor control servo and generate motor commands needed to follow the perceived motion.

2. PERCEIVING HUMAN MOTION

The key issue when realizing a real-time motion perception system is to avoid excessive interaction between pieces of data in both the temporal and spatial domains. This makes probabilistic approaches especially attractive because they allow us to govern the amount of influence of neighboring pixels in each domain by making various kinds of independency assumptions. We put the problem of body detection and tracking in a Bayesian setting, and utilize a maximum likelihood approach to find and track relevant objects in the scene. Related approaches were proposed for example in Refs. [3–5].

2.1. Tracking framework

We represent the observed environment by a number of random processes (blobs). Each entity to be tracked is represented by one process. Let us denote the probability that a pixel located at $\mathbf{u}$ having color intensity $I_{\mathbf{u}}$ was generated by the process Θ_k, $k = 1, \ldots, K$, by $\mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta_k)$. We also introduce an outlier process Θ_0, which models the data not captured by other processes. Assuming that every pixel stems from one of the mutually exclusive processes Θ_k, $k = 0, \ldots, K$ (closed-world assumption), we can write the probability that color $I_{\mathbf{u}}$ was observed at location $\mathbf{u}$

using the total probability law:

$$\mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta) = \sum_{k=0}^{K} \omega_k \mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta_k), \tag{1}$$

where ω_k is a prior (mixture) probability to observe the process Θ_k, $\sum_{k=0}^{K} \omega_k = 1$, and $\Theta = \{\Theta_0, \Theta_1, \ldots, \Theta_K\}$. Under these assumptions, the posterior probability that pixel $\mathbf{u}$ stems from the lth process is given by Bayes' rule:

$$p_{\mathbf{u},l} = \frac{\omega_l \mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta_l)}{\sum_{k=0}^{K} \omega_k \mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta_k)}. \tag{2}$$

Ignoring the correlation of assigning neighboring pixels to processes, the overall probability to observe image $\mathbf{I}$ can be approximated by:

$$\mathrm{P}(\mathbf{I}) = \mathrm{P}(\mathbf{I}|\Theta) = \prod_{\mathbf{u}} \mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta). \tag{3}$$

At each time step, we would like to determine $(\Theta_1, \ldots, \Theta_K, \omega_0, \omega_1, \ldots, \omega_K)$ so that likelihood (3) is maximized. Instead of maximizing criterion (3) directly, it is often easier to minimize its negative logarithm (log-likelihood).

Before we can minimize the log-likelihood, we must decide how to model the process distributions Θ_k. Our approach uses shape and color properties to evaluate the probability that a pixel was generated by one of these processes. Assuming that these two properties are independent of each other, we have

$$\mathrm{P}(I_{\mathbf{u}}, \mathbf{u}|\Theta_l) \sim \mathrm{p}(I_{\mathbf{u}}|\Theta_l)\mathrm{p}(\mathbf{u}|\Theta_l). \tag{4}$$

In many cases, for example when tracking body parts, the two-dimensional (2D) shape of the tracked objects is roughly ellipsoidal and can be approximated by the center of the object's image $\mathbf{x}_l$ and by the covariance matrix $\mathbf{\Sigma}_l$ of pixels contained in it. Thus the shape part of the probability that pixel $\mathbf{u}$ belongs to the lth blob can be characterized by a 2D Gaussian distribution:

$$\mathrm{p}(\mathbf{u}|\Theta_l) = \frac{1}{2\pi\sqrt{\det(\mathbf{\Sigma}_l)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_l)\mathbf{\Sigma}_l^{-1}(\mathbf{x} - \mathbf{x}_l)\right). \tag{5}$$

Assuming that the object's texture consists of a finite number of colors, we can model the color probabilities by a Gaussian mixture model:

$$\mathrm{p}(I_{\mathbf{u}}|\Theta_l) = \sum_{k=1}^{K_l} \omega_{l,k} \mathrm{p}\big(I_{\mathbf{u}}|\overline{I_{l,k}}, \mathbf{\Gamma}_{l,k}\big), \tag{6}$$

where $\sum_{k=1}^{K_l} \omega_{l,k} = 1$ and:

$$\mathrm{p}\big(I_{\mathbf{u}}|\overline{I_{l,k}}, \mathbf{\Gamma}_{l,k}\big) = \frac{1}{\sqrt{(2\pi)^{2 \text{ or } 3}\det(\mathbf{\Gamma}_{l,k})}} \exp\left(-\frac{1}{2}\big(I_{\mathbf{u}} - \overline{I_{l,k}}\big)\mathbf{\Gamma}_{l,k}^{-1}\big(I_{\mathbf{u}} - \overline{I_{l,k}}\big)\right). \tag{7}$$

We experimented with the HSV and RGB color space, and mainly used the three-dimensional (3D) RGB space for skin color detection and tracking and the 2D HSV space (with the intensity component ignored to increase the robustness against the variations in lighting conditions) for the detection and tracking of colored objects. The outlier process is modeled by a fixed uniform distribution.

The color models are kept constant in the current version of the tracker. They are learnt off-line. Thus at each tracking step we need to maximize (3) over shape parameters $\{(\mathbf{x}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^K$ and mixture probabilities $\{\omega_k\}_{k=0}^K$. A good iterative approach is provided by an EM-algorithm, in which this is done by first calculating the posterior probabilities $p_{\mathbf{u},l}$ (given by (2), (4), (5) and (7)) using the current estimate for $\{\Theta_k\}$ and $\{\omega\}_k$ (the expectation step) and then estimating the parameters $\{(\mathbf{x}_k, \Sigma_k)\}$ and $\{\omega_k\}$ as if $p_{\mathbf{u},l}$ were constants independent of them (the maximization step). The maximization step consists of calculating the weighted mean and covariances of image pixels with $p_{\mathbf{u},l}$ being used as weights and of the reestimation of $\{\omega_k\}$. This process is repeated until convergence.

To make the real-time operation of the tracking system possible, we implemented it using techniques such as affine warping and multithreading. This implementation is described elsewhere [6]. The current version of the tracker is capable of following up to 10 blobs at 60 Hz on a dual-processor 933 MHz Pentium III computer. Since we are interested in estimating 3D blob positions, we also implemented a stereo version of the tracker. The stereo version can simultaneously track up to four blobs at 60 Hz on such a computer. We could use two PCs to simultaneously track a larger number of blobs in stereo.

2.2. Fast detection of objects of interest

Automatic detection of objects of interest and the subsequent initialization of the tracker is a very necessary part of an interactive system. As we are interested in dynamic scenes captured by the robot's eyes, it is necessary that the automatic initialization is at least as fast as the tracking algorithm. It is useless to come up with a result after a long analysis of one image because the object of interest or cameras might move to a different location before the processing is finished. In addition, the automatic detection algorithm should not require the setting of various parameters for different scenes and objects because this is tedious.

The ground knowledge for our system is provided by color and shape probability distributions. However, we use color only as ground knowledge to initialize the tracker because it is time consuming to search for objects of an ellipsoidal shape in an image. More advanced features were proposed in the literature, especially for the case of face detection [7, 8], but such an approach would require us to make much stronger assumptions about the observed objects and would thus makes the system less practical.

Based on color, the probability that a pixel belongs to the lth blob is given by (6). Since we do not have any information about the initial state of the blobs, we randomly select their shapes and locations in the image. The shape parameters

are varied in a controlled way so that 2D sizes of the generated blobs remain within prespecified limits. To achieve real-time operation, we warp a region of interest around each of the blobs onto a window of fixed size. Color probabilities (6) are then estimated at each pixel of the warped image. If the sum of all probabilities within the window exceeds a certain threshold, i.e.

$$\mathrm{f}(\Theta_l) = \sum_{\mathbf{u}} \mathrm{p}(I_{\mathbf{u}}|\Theta_l) > r_l, \tag{8}$$

the region is deemed interesting and the tracker is started using the associated, randomly selected blob parameters.

It is not possible to select thresholds r_l in advance because they depend on lighting conditions, the variability of color within the object and the quality of the color models. For example, randomly searching through a video stream for 5 s (300 images) while looking for eight objects of different colors resulted in the average and maximum probability sums that are shown in Table 1. Analyzing these results it becomes clear that we cannot set a single threshold that would account for all possible situations. Therefore we break the initialization process into two phases. First we explore the video stream for a sufficient period of time (typically for 5 s) and sample the sums from (8). We can then set a threshold for each of the sought objects to:

$$r_l = (1 - \lambda)\mathrm{E}(\mathrm{f}(\Theta_l)) + \lambda \max_{t_i} \mathrm{f}(\Theta_l), \quad 0 < \lambda < 1. \tag{9}$$

λ was typically set to 0.67 in our experiments.

In the second phase we restart the random search. The lth object is deemed found once the automatically selected threshold r_l is exceeded. If the tracker loses the object, we restart the initialization process either with the first or with the second phase depending on the application. Hence our system is also able to recover from failure.

Table 1.
Average and maximum sum of color probabilities for eight blobs over a period of 5 s

$\mathrm{E}(\mathrm{f}(\Theta_l))$	$\max_{t_i} \mathrm{f}(\Theta_l)$
1.925e−003	1.246e−001
3.956e−004	2.206e−002
5.121e−004	3.229e−002
4.852e−003	5.599e−001
6.975e−003	1.433e−001
2.329e−005	1.044e−003
1.944e−004	2.162e−002
1.245e−003	2.718e−002

2.3. 3D position estimation

We use stereo to estimate the position of tracked entities such as the hands or head. We could take the center of blobs in both images to estimate the 3D position of a tracked body part. However, the two centers give only a rather crude stereo correspondence because the blobs found in the two images do not cover the same areas on the body. This happens because of differences in the viewing direction and because of uncertainties in the estimation of shape.

Cross-correlation is a standard method for the calculation of stereo correspondences. In our case, we are not interested in generating a full depth map, but only in estimating a 3D blob position. We take the center of the blob in the left image as a starting point. A box template around the blob center is extracted and we attempt to find the best match in the right image using zero mean normalized cross-correlation:

$$\mathrm{ZNCC}_{l,r}(\mathbf{u}, \mathbf{u}+\mathbf{d}) = \frac{\mathrm{cov}(I_{\mathbf{u},l},\, I_{\mathbf{u}+\mathbf{d},r})}{\sqrt{\mathrm{var}(I_{\mathbf{u},l})}\sqrt{\mathrm{var}(I_{\mathbf{u}+\mathbf{d},r})}}, \tag{10}$$

where:

$$\mathrm{cov}(I_{\mathbf{u},l},\, I_{\mathbf{u}+\mathbf{d},r}) = \sum_{\Delta\in\mathbf{T}} \left(I_{\mathbf{u}+\Delta,l} - \overline{I_{\mathbf{u},l}}\right)^T \left(I_{\mathbf{u}+\mathbf{d}+\Delta,r} - \overline{I_{\mathbf{u}+\mathbf{d},r}}\right)/(n-1),$$

$$\mathrm{var}(I_{\mathbf{u}}) = \sum_{\Delta\in\mathbf{T}} \left\| I_{\mathbf{u}+\Delta} - \overline{I_{\mathbf{u}}} \right\|^2/(n-1),$$

and $\overline{I_{\mathbf{u}}}$ is the mean color within the box around pixel $\mathbf{u}$. The maximum of correlation (10) is sought for in a region defined by a slice of the image along the epipolar line that lies within the right blob.

3. GENERATION OF HUMANOID MOTION FROM HUMAN MOTION

To enable a robot to mimic the perceived motion, we need to map the observed trajectories into a robot-centered coordinate space. Besides the positional information, it is necessary to estimate also the velocity and acceleration of the observed motion in order to ensure that the imitated motion is as close as possible to the original human motion. The velocities and accelerations are estimated by a Kalman filter described below. This is different from an earlier version of our system in which we simultaneously estimated larger parts of an observed trajectory using splines [9]. In the previous version, a number of measurements of a motion trajectory were collected first (typically 30 measurements for 1 s of motion) and then the trajectory was estimated using spline approximation, which resulted in smooth mimicking movements. *DB* could therefore start mimicking human motion only after a delay of 1 s. In contrast to this, the current system can start tracking the observed motion immediately, but the commanded motion is based only on previous measurements and cannot take into account the future measurements.

The translation of the observed motion into a robot-centered coordinate system is simple. At the beginning of the mimicking task we save the initial position of a tracked human body part and the initial position of the corresponding robot body part, typically a hand or a face. Motion is then measured and generated with respect to these original positions. When mimicking hand motion, for example, we move the robot hand in such a way that the position of the robot hand with respect to the initial robot hand position is the same as the position of the human hand with respect to the initial human hand position.

Data provided by the vision system is given in the robot eye coordinate system. We keep the position of the eyes with respect to each other constant so that the stereo calibration remains valid during the execution of the mimicking task. The current version of the system also tries to keep the position of the robot head with respect to the robot base constant. This is achieved by exploiting the redundancy of the humanoid robot. Thus the world coordinate system remains aligned with the eye coordinate system and the estimated relative human motion can be used for the generation of a humanoid robot motion. Alternatively, we could try and compensate for the robot head motion using either our knowledge about *DB*'s kinematics or some sort of online learning.

Our vision system works at 30 or 60 Hz. Hence any motion estimated by the vision system is inevitably given at the same rate. On the other hand, *DB*'s servo rate is 420 Hz. We employ a simple PD controller in Cartesian space to generate in between points on the trajectory (positions, velocities and accelerations) and thus account for the differences in the vision frame rate and the robot servo rate. The resulting 420-Hz Cartesian motion is finally transformed into the robot motion using an inverse kinematics solver [10]. This algorithm is capable of solving the inverse kinematics of a redundant humanoid robot at servo rate. We chose to estimate not only velocities, but also accelerations, because specifying the Cartesian space accelerations ensures that the robot hand moves in the proper direction provided the estimated accelerations are correct, thus making the mimicking motion more faithful. Additionally, the computation of accelerations gives us the possibility to use control schemes based on inverse dynamics model.

A different solution to the human-humanoid mapping is presented in Ref. [11]. Unlike our approach, these authors assume less knowledge about the humanoid kinematics and make several simplifying assumptions to directly convert the Cartesian space motion into the robot joint space motion.

3.1. Kalman filter

In this section we present the estimation of human motion parameters based on the 3D blob positions provided by the vision system. The kinematics of any point **P** belonging to a rigid body is completely characterized by $\mathbf{v}(t)$, the velocity of the point on the body coinciding with the origin **O** of the body-fixed coordinate system, and $\omega(t)$, the angular velocity of the point **P** around the origin of the reference

coordinate system:

$$\mathbf{v_P}(t) = \mathbf{v}(t) + \omega(t) \times \overrightarrow{\mathbf{OP}}. \tag{11}$$

A closed form solution for this general motion is not available. It is possible to solve (11) in some special cases, but the solution would be quite different for different situations because forces acting on the human body are very much dependent on the current situation. Since we do not wish to make any specialized assumptions about the dynamics of the observed motion, we selected a probabilistic approach to approximate any kind of human motion. To enable the estimation of the body part's position, velocity and acceleration based on the Kalman filter, we assume that the linear jerk is a stationary random process $\phi(t)$ with the following properties:

$$\mathrm{E}(\phi(t)) = 0, \quad \mathrm{E}(\phi(t)\phi(t+s)^T) = b^2\delta(s)\mathbf{I}, \tag{12}$$

where $\mathbf{I}$ denotes the identity matrix, δ is the Dirac delta function and b^2 is the spectral amplitude of ϕ. Note that we assume that different components of the random jerk vector are uncorrelated and equally large. In contrast to a more common random acceleration model, the random jerk model enables us to estimate not only positions and velocities but also accelerations. This is important because as explained above, it is beneficial for the mimicking system to have the accelerations available.

At every measurement time t_i we estimate the state vector $\mathbf{x}(t) = [\mathbf{p}(t)^T, \mathbf{v}(t)^T, \mathbf{a}(t)^T]^T$, which consists of the object's position, linear velocity and linear acceleration. Since previously published systems usually employ the random acceleration instead of the random jerk model, we describe the estimation procedure in detail below. The random jerk model yields the following equation of motion

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \phi(t) \end{bmatrix}. \tag{13}$$

The orientation of body parts is ignored in the current version of the system, but see Refs. [12, 13] for the treatment of a complete rigid body motion estimation problem.

The integration of (13) results in the following state propagation equation, which is suitable for use in a Kalman filter framework [14]:

$$\mathbf{x}(t_{i+1}) = \mathbf{A}(t_{i+1}, t_i)\mathbf{x}(t_i) + \mathbf{w}_i, \tag{14}$$

where:

$$\mathbf{A}(t, s) = \begin{bmatrix} \mathbf{I} & (t-s)\cdot\mathbf{I} & (t-s)^2/2\cdot\mathbf{I} \\ 0 & \mathbf{I} & (t-s)\cdot\mathbf{I} \\ 0 & 0 & \mathbf{I} \end{bmatrix} \tag{15}$$

and:

$$\mathbf{w}_i = \int_{t_i}^{t_{i+1}} \mathbf{A}(t_{i+1}, t)[\,0 \quad 0 \quad \psi(t)\,]^T \,\mathrm{d}t. \tag{16}$$

Using (12) and (16), we can estimate the process noise covariance, i.e. the covariance of $\mathbf{w}_i$, as follows:

$$\begin{aligned}\mathbf{Q}_i &= \mathrm{E}\left(\mathbf{w}_i \mathbf{w}_i^T\right) \\ &= \mathrm{E}\left(\int_{t_i}^{t_{i+1}} \mathbf{A}(t_{i+1}, t)[\,0 \quad 0 \quad \phi(t)^T\,]^T \,\mathrm{d}t \int_{t_i}^{t_{i+1}} [\,0 \quad 0 \quad \phi(s)^T\,]\mathbf{A}(t_{i+1}, s)^T \,\mathrm{d}s\right) \\ &= b^2 \begin{bmatrix} (t_{i+1}-t_i)^5/20 \cdot \mathbf{I} & (t_{i+1}-t_i)^4/8 \cdot \mathbf{I} & (t_{i+1}-t_i)^3/6 \cdot \mathbf{I} \\ (t_{i+1}-t_i)^4/8 \cdot \mathbf{I} & (t_{i+1}-t_i)^3/3 \cdot \mathbf{I} & (t_{i+1}-t_i)^2/2 \cdot \mathbf{I} \\ (t_{i+1}-t_i)^3/6 \cdot \mathbf{I} & (t_{i+1}-t_i)^2/2 \cdot \mathbf{I} & (t_{i+1}-t_i) \cdot \mathbf{I} \end{bmatrix}. \end{aligned} \tag{17}$$

There is no systematical way to automatically select parameter b, the amplitude of the process noise. In our experiments, we obtained good results by setting $b = 27\,500$, where the 3D measurements were given in millimeters. This ensured a good trade-off between smoothness and goodness of fit. The guideline is that a smaller b results in more smoothing, whereas a larger b keeps the estimated 3D positions closer to the measurements.

The measurements are provided by the previously described stereo vision system. They are related to the state vector via the state measurement equation:

$$\mathbf{p}_i = \mathbf{H}\mathbf{x}_i + \mathbf{v}_i, \quad \mathbf{H} = [\,\mathbf{I} \quad 0 \quad 0\,], \tag{18}$$

where $\mathbf{v}_i$ denotes the white measurement noise. It is important that we properly estimate the covariance matrix of the measurement noise vector because, as it is well known, the stereo triangulation errors are oriented and depend on depth. Nearby points have a fairly compact uncertainty, whereas distant points have a more elongated uncertainty that is roughly aligned with the viewing direction towards the point [15]. It is common to assume that the 2D image noise is constant and to estimate the 3D covariance matrix $\mathbf{R}_i$ of $\mathbf{p}_i$ using standard covariance propagation rules. Table 2 shows that this approach gives an accurate approximation for the true measurement covariance. In our experiments, the 2D covariances were estimated in advance from the data. These 2D matrices were then used to estimate the 3D position covariances depending on the blob position in space.

Table 2.

$\begin{bmatrix} 6.5\text{e}{-1} & -2.3\text{e}{-2} \\ -2.3\text{e}{-2} & 6.7\text{e}{-1} \end{bmatrix}$	$\begin{bmatrix} 6.6\text{e}{-1} & -1.2\text{e}{-2} \\ -1.2\text{e}{-2} & 6.7\text{e}{-1} \end{bmatrix}$
$\begin{bmatrix} 4.4\text{e}{+1} & -1.6\text{e}{+1} & -2.4\text{e}{+2} \\ -1.6\text{e}{+1} & 9.4\text{e}{+0} & 9.3\text{e}{+1} \\ -2.4\text{e}{+2} & 9.3\text{e}{+1} & 1.4\text{e}{+3} \end{bmatrix}$	$\begin{bmatrix} 4.3\text{e}{+1} & -1.5\text{e}{+1} & -2.3\text{e}{+2} \\ -1.5\text{e}{+1} & 8.5\text{e}{+0} & 8.8\text{e}{+1} \\ -2.3\text{e}{+2} & 8.8\text{e}{+1} & 1.4\text{e}{+3} \end{bmatrix}$

In this experiment we took 600 images of an object at rest. The two 2×2 matrices in the upper row show the average covariances of the detected 2D blob positions that were estimated from the data. The left 3×3 matrix in the lower row shows the average covariance of the 3D blob position estimated from the 3D data. The right 3×3 matrix in the lower row shows the average covariance obtained by transforming the 2D covariances using standard covariance propagation rules.

Vision can directly measure only positions in space. Hence once a body part is detected and the tracking starts, we can only acquire those coordinates in the state vector that correspond to the blob position. The associated part of the state covariance matrix is simply taken to be the current measurement covariance. As we have no information about the initial velocities and accelerations, these values are initially set to zero. The associated part of the state covariance matrix is set to a diagonal matrix with very large elements on the diagonal, reflecting the fact that we have little confidence in this initial approximation. This enables the Kalman filter to quickly adapt the corresponding parameters to the perceived motion.

This completes the specification of all entities needed to run the Kalman filter. We made use of the OpenCV [16] implementation of the filter to carry out the filtering process.

4. EXPERIMENTS

The methods described in the previous two sections enable us to detect a human body part, track its motion, estimate its position, velocity and acceleration, and to mimic the estimated motion by a humanoid robot. The result is an immediate and realistic imitation of the perceived motion. In this section we presents some results showing the accuracy of the presented methods.

To show that our system returns meaningful results, we first measured the 3D position of a human hand which was not moving. The raw results and the filtered data are shown in Table 3. It is clear from these results that the Kalman filter succesfully reduced the noise in the vision data and that the estimated velocities and accelerations were closed to the true value (0). As expected, the noise and consequently the smoothing effects are larger in the z direction which is aligned with the optical axis of the left eye (the direction of depth).

In the second experiment the human performer tried to alternatively move his hand as much as possible along the three coordinate axes of the robot's base coordinate system. Figure 1 shows the trajectory estimated by the proposed Kalman filter technique. Motion along the coordinate axes is clearly reflected not only in the position trajectories, but also in the velocities and accelerations. Again, the noise in the direction of the optical axis is larger than the noise in other directions. It is also not surprising that the noise in velocities and accelerations is larger than the noise in the measured positions. These results were acquired during the execution of the mimicking task.

Table 3.
Accuracy of the filter (mm) when estimating the motion of an object that does not move (mean (SD))

Raw position	Position	Velocity	Acceleration
−88.9 (2.96)	−88.7 (1.82)	−0.27 (25.02)	−2.34 (200.25)
221.1 (7.92)	219.9 (3.93)	0.78 (28.30)	2.37 (164.59)
1024.6 (35.00)	1019.2 (16.80)	3.71 (91.78)	12.88 (263.88)

Figure 2 shows that *DB* can quite accurately follow the commanded trajectory. The executed motion is close to the desired motion, although slightly less smooth. The main problem when mimicking the performed human motion is that *DB*'s joint limits are much stricter than human joint limits. *DB*'s motion becomes more jerky when *DB* approaches his joint limits. Another inaccuracy is caused by the very wide lenses used for *DB*'s eyes. This makes distortion effects significant once we approach the edge of an image. Although we developed a method that can correct the radial distortion effects, these effects are still significant enough to influence the quality of the measurements even after the distortion correction. This can also cause a rather jerky motion.

In general, it is easier to mimic movements that do not involve motion in the direction of depth because the estimation of motion parameters in this direction is much more noisy (see Fig. 1 and Table 3). If the depth does not change, the Kalman filter can easily eliminate the noise in this direction. On the other hand, it is difficult for the filter to distinguish noise from real motion if the depth changes rapidly. The resulting motion is therefore jerkier in this case. To some extent this problem can be alleviated by tuning the spectral amplitude b^2 from the random jerk model (12), but there is no automatic method to accomplish this task.

Finally, in Figs 3 and 4 we present a few images showing *DB* successfully following human motion. We carried out various experiments, among them mim-

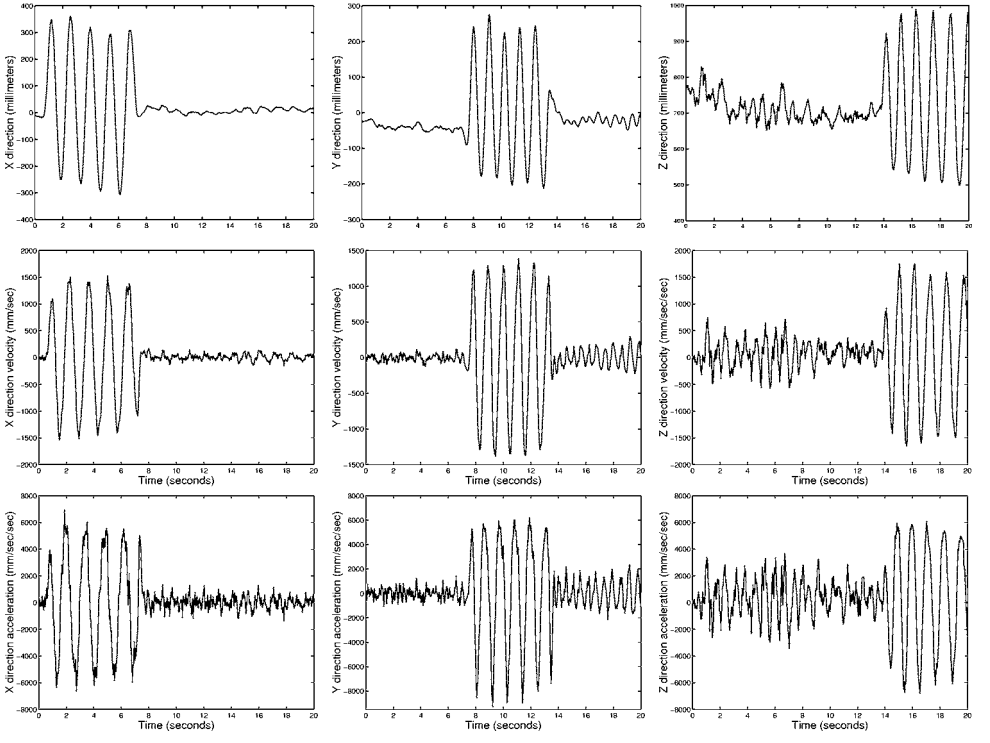

Figure 1. Positions, velocities and accelerations estimated by the Kalman filter.

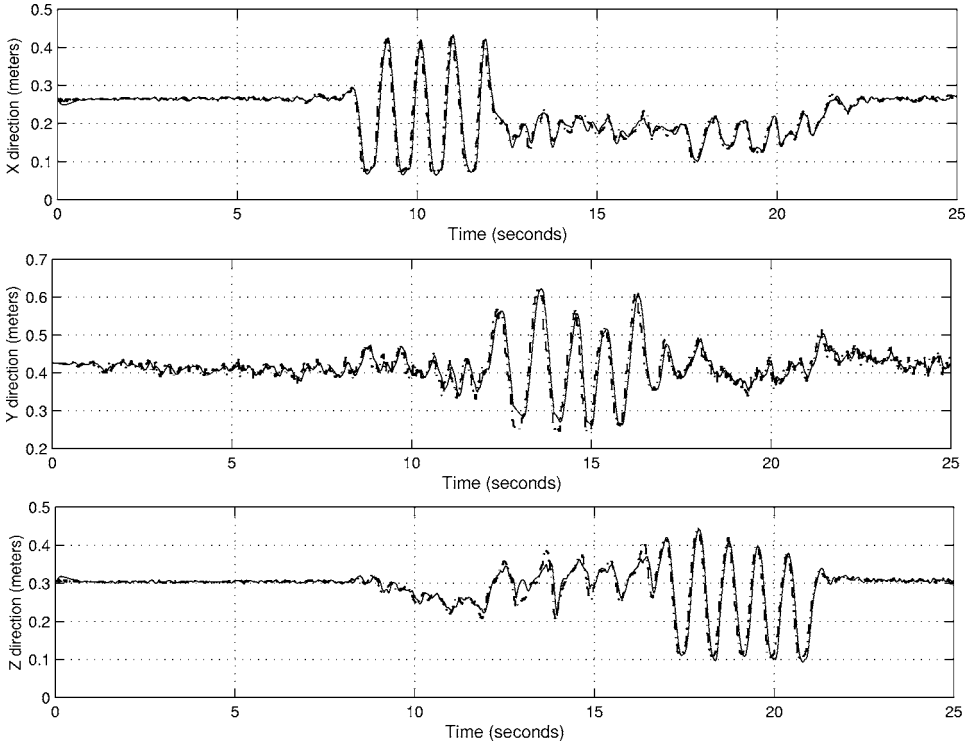

Figure 2. The estimated Cartesian space motion trajectory (solid) and the trajectory followed by the robot (dash dot).

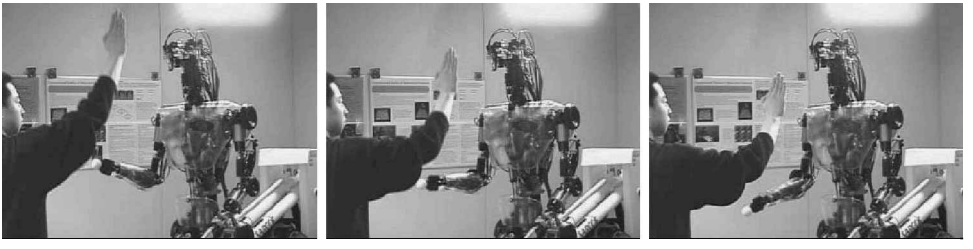

Figure 3. *DB* mimicking downwards hand motion.

Figure 4. *DB* tracking left to right head motion with his right hand.

icking of human hand motion and tracking of human head motion with *DB*'s hand.

5. CONCLUSIONS

We have presented an integrated vision and motor control system that enables a humanoid robot to mimic human motion in real-time. The vision system was entirely implemented on a standard PC and can thus be easily replicated. The developed system represents an important building block for higher-level imitation learning systems that can use visual information provided by the robot's eyes and general vision.

In addition to studying higher level imitation learning and interaction issues, we still need to deal with a number of immediate problems that affect the operation of the current system. The most urgent problem is the compensation of the head motion during the execution of mimicking tasks. We intend to use learning to compensate for image motion generated by the robot.

Another important issue is the choice of a body configuration. For example, a humanoid arm is redundant with respect to the mimicking of hand motion and there are an infinite number of configurations that result in the same hand motion. In the experiment in Fig. 3, the robot arm configuration was different from the demonstrator's arm configuration when the mimicking began and this difference was retained throughout the mimicking session. Ideally, the visual system should recognize the initial arm configuration, but this is a formidable task for visual processing, especially if it is to be performed in real-time. An off-line approach is described in Ref. [17], but we are currently working on a real-time solution.

Acknowledgements

Most of this work was done at ATR, Kyoto, Japan, in the frame of the CyberHuman Project. Support for C. G. A. was also provided by US National Science Foundation Award IIS-9711770. Support for A. U. was also provided by the Slovenian Ministry of Education, Science and Sport.

REFERENCES

1. C. G. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar and M. Kawato, Using humanoid robots to study human behavior, *IEEE Intell. Syst.* **15** (4), 46–56 (2000).
2. S. Schaal, Is imitation learning the route to humanoid robots? *Trends Cognitive Sci.* **3** (6), 233–242 (1999) .
3. N. Jojic, M. Turk and T. S. Huang, Tracking self-occluding articulated objects in dense disparity maps, in: *Proc. IEEE Int. Conf. Computer Vision*, Kerkyra, Greece, pp. 123–130 (1999).
4. S. J. McKenna, Y. Raja and S. Gong, Tracking colour objects using adaptive mixture models, *Image Vision Comput.* **17** (3-4), 225–231 (1999).

5. C. R. Wren, A. Azarbayejani, T. Darell and A. P. Pentland, Pfinder: real time tracking of the human body, *IEEE Trans. Pattern Anal. Machine Intell.* **19** (7), 780–785 (1997).
6. A. Ude and C. G. Atkeson, Probabilistic detection and tracking at high frame rates using affine warping, in: *Proc. Int. Conf. Pattern Recognition*, Québec, Vol. 2, pp. 6–9 (2002).
7. H. Rowley, S. Baluja and T. Kanade, Neural network-based face detection, *IEEE Trans. Pattern Anal. Machine Intell.* **20**, 22–38 (1998).
8. P. Viola and M. Jones, Robust real-time object detection, Tec. Report CRL 2001/01, Compag Cambridge Research Laboratory. Cambridge, MA (2001).
9. A. Ude and C. G. Atkeson, Real-time visual system for interaction with a humanoid robot, in: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Maui, HI, pp. 746–751 (2001).
10. G. Tevatia and S. Schaal, Inverse kinematics for humanoid robots, in: *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 294–299 (2000).
11. G. Cheng and Y. Kuniyoshi, Real-time mimicking of human body motion by a humanoid robot, in: *Proc. 6th Int. Conf. Intelligent Autonomous Systems*, Venice, Italy, pp. 273–280 (2000).
12. D. B. Gennery, Visual tracking of known three-dimensional objects, *Int. J. Comput. Vis.* **7** (3), 243–270 (1992).
13. A. Ude, Filtering in a unit quaternion space for model-based object tracking, *Robotics Autonomous Syst.* **28** (2-3), 163–172 (1999).
14. Y. Bar-Shalom and T. F. Fortmann, *Tracking and Data Association.* Academic Press, New York (1988).
15. L. Matthies and S. A. Shafer, Error modeling in stereo navigation, *IEEE J. Robotics Automat.* **3** (3), 239–248 (1987).
16. Open Source Computer Vision Library (OpenCV). //http://www.intel.com/research/mrl/research/opencv/
17. A. Ude, Robust estimation of human body kinematics from video, in: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Kyongju, Korea, pp. 1489–1494 (1999).

ABOUT THE AUTHORS

Aleš Ude studied Applied Mathematics at the University of Ljubljana, Slovenia, and Computer Science at the University of Karlsruhe, Germany, where he received a Doctoral degree in 1996. From 1998 to 2000 he was an STA fellow in the Kawato Dynamic Brain Project, ERATO, JST. Currently he holds a research position at the Jožef Stefan Institute, Ljubljana, Slovenia, and is also associated with the CyberHuman Project, ATR-I, Kyoto, Japan. His research interests include the visual perception of human activity and its application to robot learning.

Christopher G. Atkeson is an Associate Professor at the Robotics Institute and the Human–Computer Interaction Institute at Carnegie Mellon University. His research focuses on numerical approaches to machine learning, and uses robotics and intelligent environments as domains in which to explore the behavior of learning algorithms. He is a recipient of a National Science Foundation Presidential Young Investigator award.