Real-Time Hand Tracking with a Color Glove for the Actuation of Anthropomorphic Robot Hands

Matthias Schröder[1], Christof Elbrechter[2], Jonathan Maycock[2], Robert Haschke[2], Mario Botsch[1], Helge Ritter[2]
[1]Computer Graphics & Geometry Processing Group, [2]Neuroinformatics Group,
Bielefeld University, Germany

***Abstract*—We extend a recent low cost real-time method of hand tracking and pose estimation in order to control an anthropomorphic robot hand. The approach is data-driven and based on matching the current image of a color-gloved hand with the best fitting image in a database to retrieve the posture. Then, using depth information from a Kinect camera and a color-sensitive iterative closest point-to-triangle algorithm we can very accurately estimate the absolute position and orientation of the hand. The effectiveness of the approach is demonstrated in an application in which we actively control a 20 DOF anthropomorphic robot hand in a manual interaction grasping task.**

I. Introduction

Our goal was to create a low cost real-time hand tracking and pose estimation system to control anthropomorphic robotic hands. Our system allows for the fast generation of real hand posture data that can be used in, for example, teaching by demonstration scenarios or in the direct control of anthropomorphic robotic hands.

Most existing methods that track and estimate the posture of the hand are either too slow to be used in interactive real-time applications or too bulky and expensive. We use a data-driven approach in which a color glove is detected and using a nearest neighbor search in an image database the closest matching image and corresponding posture and coarse rotation are retrieved. Our approach, following that of Wang and Popović [1], is a low cost real-time system that can provide accurate hand posture and position estimation. A simplification of their algorithm along with the addition of a Kinect [2] camera distinguishes both approaches. Whereas Wang and Popović's approach was concerned with producing accurate hand postures, our primary focus is on accurately estimating the pose (position and orientation) of the hand. In order to perform robot actuation in tasks such as grasping, controlling the pose of the hand takes on greater importance than having an accurate posture, especially when using, as we are, compliant robot hands. The Kinect camera provides a 3D point cloud, which we use to initially place the hand and then optimize this position using an Iterative Closest Point-to-Triangle (ICP-T) algorithm.

II. Related Work

Hand tracking and especially hand posture estimation is a difficult problem due to the complexity of the hand itself, which has, excluding position and orientation, 27 degrees of freedom [3]. This has resulted in many researchers turning to either vision systems that require markers to be placed on the hands or data-gloves [4].

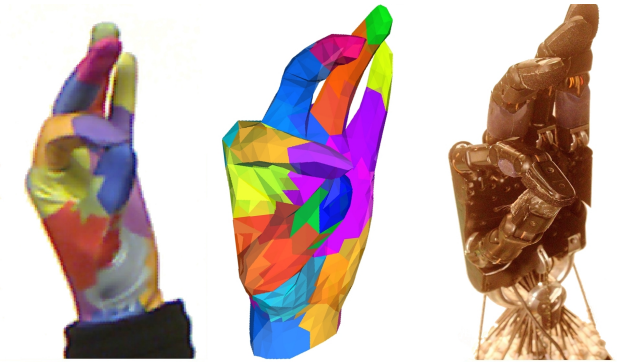

Fig. 1. The color glove is used to facilitate hand posture estimation. The resulting posture and pose are used to control an anthropomorphic robot hand in real-time.

Optical tracking systems that employ markers, such as the VICON system, are able to capture highly accurate kinematic movement data of the hand [4], [5]. However, these approaches can suffer from occlusions of one or more markers, which can occur especially when grasping and manipulating objects, and have the disadvantage that usually an expensive and large optical system is required. An alternative is to use data-gloves that measure joint angles directly and this method was extensively used to control robot hands in the past [6], [7], [8]. However, data-gloves with embedded sensors can be quite bulky to wear and thus impede natural movements. Furthermore, they generally do not provide position and orientation information, and the mapping of raw sensor values to joint angles requires non-trivial calibration procedures [9].

Bare hand tracking is desirable for certain applications, and recently existing model-based approaches that heretofore had proved too computationally expensive for real-time applications [10] are now becoming feasible. Oikonomidis et al. have introduced two such approaches, one using a multi-camera setup [11] and the other using the Kinect camera [12]. While these are very promising directions, they suffer from the high computational complexity that is associated with model-based approaches and therefore had to be optimized to run on a GPU. The Kinect camera was used for hand and finger detection in [13], where several hand gestures were recognized to facilitate real-time interaction in a graphical

user interface application. In our work, we provide a more complete estimate of the full kinematic state of the user's hand.

As a compromise between approaches based on markers or data gloves and bare-hand tracking approaches, Wang and Popović [1] devised a method for real-time hand tracking and posture estimation using a color glove. A camera captures images of the glove and after segmentation and normalization the closest matching image in a large database, along with its unique posture, is found. A database of synthetically generated images is used for directly matching the detected features instead of performing a complex search in the hand configuration space. This data-driven approach simplifies the hand tracking problem considerably and is able to provide good posture and orientation information. Use of a color glove improves detection, feature extraction and speed over approaches that use bare hands. In [14] Wang et al. adapted their method to facilitate markerless 6D hand pose estimation using a dual camera setup in a CAD application scenario. To cope with the drawbacks of using only images of bare hand silhouettes instead of color images, several restrictions to the user's freedom of movement had to be introduced in this work. While these restrictions were acceptable in a CAD context, they are not in our case. We use the color glove-based work by Wang and Popović as the foundation of our hand tracking approach, however we have made a number of simplifications to their algorithm and have introduced a Kinect camera in order to be able to much more precisely control the absolute position and orientation of the hand in 3D space. This is crucial if robust control of anthropomorphic robot hands is to be achieved.

The task of hand tracking for robot control was previously dealt with in [15], where Do et al. employed a data-driven approach for grasp recognition and hand pose estimation and mapped the estimation results to a humanoid robot. In their method [16] features extracted from bare hand images are matched in a synthetically generated database to obtain the posture and orientation of the hand. This represents a discriminative hand pose estimation approach, whereas our approach combines discriminative and generative estimation by fitting a virtual hand model to the Kinect point cloud after obtaining the initial posture estimate from a database, making use of the glove's unique color pattern in both estimation steps.

III. Hand Tracking Method

The color glove was designed by Wang and Popović with a pattern of 20 patches in 10 distinguishable colors (see Fig. 1). We make use of its distinctive pattern to efficiently find the current image's closest match in the database and thus retrieve the hand posture and a coarse estimation of the rotation of the hand. Aligning the color image with the depth values provided by the Kinect camera allows us to accurately estimate the global position and orientation of the hand. Combining the posture (joint angles) and the pose (position and orientation) allows us to control an anthropomorphic robot hand.

Our approach relies on a database populated with images of a virtual color glove along with their associated posture and rotation information. The depth information from the Kinect camera is currently only used to optimize the pose of the hand and therefore is not contained in the database. The first step was to capture natural realistic postures performed by one of the authors and then to map these onto a kinematic hand model. The model used has 22 joint angles, specifically 16 flexion angles and 6 abduction angles for each of the finger bases and the wrist joint [4]. This simplified hand model matches the kinematics of the shadow robot hand well and makes the task of mapping joint angles to the robot easier. Ground truth data was captured using a Vicon motion tracking system [17] to track reflective markers placed on a human hand as it performed various movements. After converting the raw positional Vicon data into joint angles, we sub-sampled the postures using a low-dispersion sampling method [1] and constructed several task-specific posture databases (see Sec. IV), varying the number and type of postures contained within them.

Each database contains a number of posture entries. The joint angles for each posture were used to animate a virtual hand model textured with the glove's color pattern. The hand model was then rendered from multiple camera views and each database posture entry was indexed by these different views. The camera positions were produced by uniformly sampling a virtual sphere surrounding the hand model, which was accomplished by performing a uniform subdivision of an icosahedron to approximate the surrounding sphere and placing the cameras at its vertices. At each position, the camera was rotated around the optical axis in several steps to encode the rotation of the color glove in the image plane. We used 42 camera positions with 18 rotations around the optical axis each, resulting in a total of 756 views per posture in all of our databases. Finally, for each posture and view entry, a number of tiny images of various sizes were created. Tiny images [1] are simply normalized downscaled color-classified images of the color glove. We generated several tiny image sets using image dimensions ranging from 8×8 pixels up to 64×64 pixels. These are used to improve efficiency when searching for the best match in the database for the current camera image of the hand.

Figure 2 provides an overview of our system. The three main parts of the system that produce as output a final hand posture and pose are introduced in the following sections.

A. Color-Labeled Point Cloud Creation

The data obtained from the Kinect is processed to create a 3D point cloud containing the positional and color information of the detected glove. The camera parameters of the Kinect's color and depth cameras were obtained in a stereo calibration procedure and are used to perform an RGBD-mapping, which maps color values to the pixel coordinates of the depth image D. After applying this mapping, the

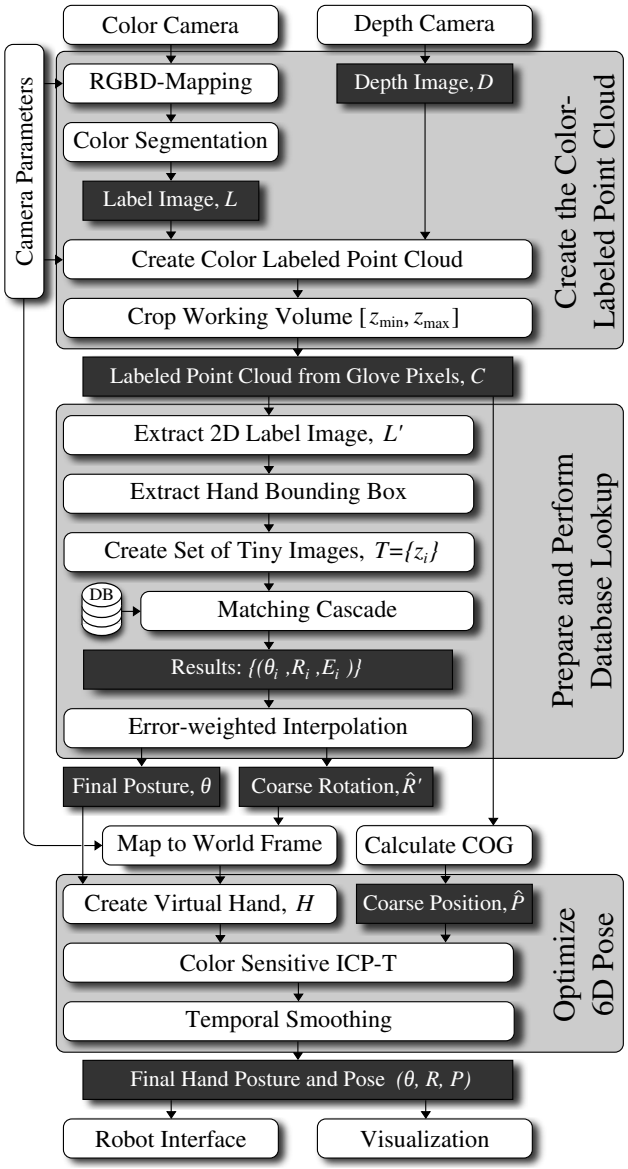

Fig. 2. Overview of the system.

color glove is detected in the mapped image, M, and using a lookup table-based color segmentation procedure a color labeled image, L, is created. The color segmentation is performed in the YUV color space to reduce sensitivity to changes in lighting. Each of the glove's ten unique colors are labeled with numbers 1 to 10 and colors that do not belong to any known color class are labeled 0. Now we are in a position to compute a color-labeled 3D point cloud

$$C = \left\{ (p_{ij}, L_{ij}) \;\middle|\; L_{ij} \neq 0, p_{ij_z} \in [z_{min}, z_{max}] \right\}, \quad (1)$$

where p_{ij} is the 3D position and L_{ij} the color class label for every glove pixel (i, j). To create C, we first ensure that only pixels in L that are not 0 are considered. The (x, y, z) position is computed for these pixels and in order to minimize sensitivity to color segmentation outliers only those

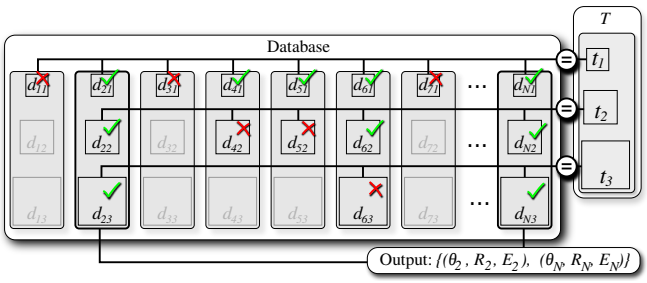

Fig. 3. The set $T = \{t_i\}_i$ of tiny images is matched successively against all N database indices d_{ji}, $j \in \{1 \;..\; N\}$. The output is a 3-tuple set $\{(\theta_j, R_j, E_j)\}_j$, where θ_j, R_j and E_j are the estimated posture, rotation and matching error, respectively.

whose z world coordinates reside in the working volume $[z_{min}, z_{max}]$ are kept.

B. Database Lookup

The output from the previous step is a labeled point cloud, C, and from this we create a new labeled image by masking C with the labeled image, L, from the RGB camera. Using a bounding box, we extract the hand from C and from this we create a set of tiny images (with dimensions ranging from 8×8 pixels up to 64×64 pixels). We need to be able to quickly compute the similarity between a tiny image in the database and one computed from the current camera image. To do this we use a Hausdorff-like distance [18] defined as

$$m(A, B) = \frac{1}{|G_A|} \sum_{a \in G_A} \min_{b \in C_a} \|a - b\|, \quad (2)$$

where $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_n\}$ are the two images, G_A is the set of glove pixel coordinates in image A, C_a is the set of pixel coordinates in image B that have the same color as the pixel at $a \in G_A$ and $\|\cdot\|$ is the Euclidean norm. We used chamfering [19], in contrast to Wang and Popović who employed similarity sensitive coding [20], to approximate Euclidean distance transforms for every color class in the images, allowing for an efficient direct lookup-based calculation of the distance between two images.

To take advantage of the varying speed and accuracy properties of differently sized tiny images, we perform a multi-stage k nearest neighbor lookup for each sized tiny image computed from the current camera image. The set $T = \{t_i\}_i$ of tiny images is used as input to the matching cascade database lookup step (see Fig. 3) and is matched successively against all N database indices d_{ji}, $j \in \{1, \ldots, N\}$. In the figure only 3 different tiny image sizes are used, i.e., $i \in \{1, 2, 3\}$. Initially, smaller tiny images, whose Hausdorff distance can be computed quickly, albeit yielding a lower accuracy, are used to efficiently pre-select nearest neighbor candidates. At each iteration, larger tiny image matching, which is computationally more expensive but results in higher accuracy, is performed to find the k_i nearest neighbors in significantly smaller sets of candidate images (in the figure $k_1 = 5, k_2 = 3, k_3 = 2$). The output is created from the

database indices that remain after the last cascade step (in the figure d_{23} and d_{N3}). This results in a set $\{(\theta_j, R_j, E_j)\}_j$, where θ_j and R_j are the posture and rotation associated with the database index d_j, and E_j is the corresponding matching error in the last cascade step. This set $\{(\theta_j, R_j, E_j)\}_j$ is then interpolated, weighting the contributions of (θ_j, R_j) pairs with their matching error E_j to the tiny image computed from the current camera image.

C. 6D Pose Optimization

The result of the database lookup is the 22-dimensional hand posture estimate, θ, along with a coarse estimate of the orientation of the hand towards the camera, $\hat{R}$. We compute the final estimate of the hand's 6D pose (rotation R and position P) by aligning a virtual 3D model of the hand to the observed point cloud. For this we perform a color-sensitive Iterative Closest Point-to-Triangle (ICP-T) computation, where the relative rigid transformation between the point cloud, C, and the surface of the virtual hand model, H, is computed and iteratively refined. By performing a color-sensitive closest point search, the observed points are only matched to model surface patches with corresponding colors. As a result, the color patterns in the observation point cloud are matched to the same color patterns on the surface of the virtual hand model during the ICP-T process. This ensures a plausible alignment of the model to the sensor data even if the posture estimation from the previous step does not fit perfectly, which cannot be accomplished when using only positional information.

As a preparation for the alignment process, the coarse rotation estimate, $\hat{R}'$, is mapped to the world coordinate frame and the center of gravity (COG) of C is computed. This yields a coarse pose estimate $(\hat{R}, \hat{P})$. We represent this pose estimate as a 4×4 transformation matrix, $\hat{T}$. The virtual hand model, H, is initialized by animating it according to the joint angles, θ, using linear blend skinning [21] and transforming it using $\hat{T}$. In each iteration of the color-sensitive ICP-T, we iterate over the points in C and for each point find the closest point on the surface of H of corresponding color. H is textured with the color glove pattern, so the color class label of each triangle in the model is known. To find the surface point closest to $(p_{ij}, L_{ij}) \in C$, we iterate through all triangles in H with the same color class L_{ij} and store the surface point, p_{ij}^H, with the minimal point-to-triangle distance [22]. We perform back-face culling beforehand to exclude triangles directed away from the sensor camera. Based on the set of corresponding points $\{(p_{ij}^H, p_{ij})\}_{i,j}$ that results from the closest point search, we calculate the 4×4 relative transformation matrix T_k from H to C in ICP-T iteration k using a common approach based on eigenvector analysis and quaternions [23]. The transformation T_k is then applied to the virtual hand model H, moving it towards the target point cloud C. After this, the alignment error is given by the mean of the new point-to-triangle distances. The ICP-T process is repeated until the alignment error converges. After convergence, the final hand pose estimate T_H is given by the product of the initial coarse transformation matrix and all K ICP-T iteration matrices:

$$T_H = \left(\prod_{k=K}^{1} T_k \right) \hat{T} = \begin{pmatrix} R & P \\ 0 & 1 \end{pmatrix}. \quad (3)$$

The output of our hand tracking system is the final hand posture and pose estimate (θ, R, P). To reduce jitter, we smooth the posture and pose estimation by interpolating with estimations from the three previous frames, which still allows for responsive tracking of dynamic movements.

Posture Error

	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$DB_{open+pinch}$	1.34	0.95	–	–	–
$DB_{open+power}$	–	–	3.42	2.60*	1.60
$DB_{open+pinch+power}$	4.89	4.04	3.23	2.57	2.17
$DB_{20\ postures}$	18.0**	10.30	7.40	8.91	8.99

Rotation Error

	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$DB_{open+pinch}$	4.98	4.06	–	–	–
$DB_{open+power}$	–	–	3.11	2.69*	3.10
$DB_{open+pinch+power}$	5.59	4.47	3.03	3.46	4.59
$DB_{20\ postures}$	11.67**	6.28	3.52	4.39	4.79

Position Error

	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$DB_{open+pinch}$	3.90	5.41	–	–	–
$DB_{open+power}$	–	–	4.72	3.88*	4.41
$DB_{open+pinch+power}$	4.64	6.01	4.65	4.04	4.82
$DB_{20\ postures}$	7.33**	7.46	4.81	4.96	6.11

Fig. 4. Posture and pose estimation errors for various ground truth experiments. Posture and rotation errors are given in deg, position errors are given in mm. For all ground truth trajectories, only databases containing the respective grasping postures were used. The trajectory of the errors labeled with superscript * is shown in detail in Figure 5, the trajectory of the errors labeled with superscript ** is shown in detail in Figure 6.

IV. Evaluation

In this section we evaluate the performance of our system in three main test scenarios. First we consider the accuracy of the posture and pose, then the effect of various different matching cascades on the database retrieval accuracy and the runtime efficiency of the database lookup, and finally the system runtime speed versus position error with various different sub-sampling factors for the ICP-T algorithm.

We use four different databases in the following tests: two 2-posture databases, $DB_{open+pinch}$ and $DB_{open+power}$ containing an open hand and a pinch and power grasp, respectively, a 3-posture database, $DB_{open+pinch+power}$, containing all three postures, and a database with 20 postures, $DB_{20\ postures}$, containing the previous 3 postures plus 17 random movement postures. Ground truth data was created by capturing five different movement trajectories: two containing a pinch grasp, an open hand and some movements (Tr_1 and Tr_2) and three containing a power grasp, an open hand and some movements (Tr_3, Tr_4 and Tr_5). These movements were tracked by our system and their computed postures and poses became the ground truth for the subsequent tests. Re-playing these movement trajectories allowed us to create synthetic ground truth Kinect data (synthetic color and depth

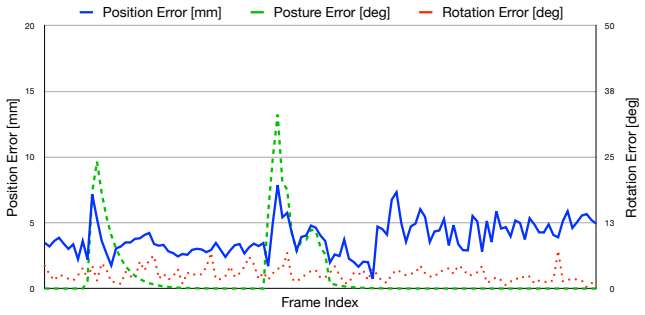

Fig. 5. Detailed overview of trajectory Tr_4 using database $\mathrm{DB}_{\text{open+power}}$. Several keyframes of the trajectory are visualized in the bottom of the figure. The upper row of images shows the virtual hand model according to the ground truth, the lower row of images shows the difference of the ground truth and estimation depth images.

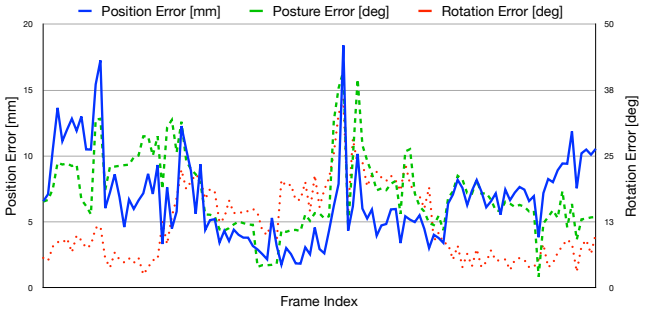

Fig. 6. Detailed overview of trajectory Tr_1 using database $\mathrm{DB}_{20\ \text{postures}}$. Analogous to Figure 5.

images), which was then used as input for our hand tracking system. Gaussian noise, whose variance was scaled with the gradient intensity image, was added to each synthetic depth frame in order to simulate real data. We were then able to directly calculate the error between the ground truth postures and poses and their estimation.

A. *Posture and pose estimation*

To evaluate the estimation quality of our system, we computed posture, rotation and position errors for all ground truth trajectories using all databases. Figure 4 shows the average errors for all experiments. Databases were only used if they contained the main grasp (pinch or power) performed in a particular movement trajectory. In most cases, the errors are lowest when using the respective small grasp-specific databases ($\mathrm{DB}_{\text{open+pinch}}$ or $\mathrm{DB}_{\text{open+power}}$) and highest with a large database containing several hand postures in addition to both grasping postures ($\mathrm{DB}_{20\ \text{postures}}$).

Figure 5 gives a detailed overview of the posture and pose errors for trajectory Tr_4 using database $\mathrm{DB}_{\text{open+power}}$. This combination resulted in a low average error in all estimation parameters. The maxima that can be observed in the posture error can be explained by a quick hand posture change in the ground truth trajectory while opening and closing the hand. The posture estimation was lagging behind the ground truth for several frames before assuming the correct posture. Figure 6 shows the posture and pose errors for trajectory Tr_1 using database $\mathrm{DB}_{20\ \text{postures}}$. In this experiment, larger errors occurred when database entries not related to the pinch grasp motion of the ground truth trajectory influenced the estimation result.

The average positional error of our hand pose estimation lies below 1cm in all of our experiments. This shows a significant improvement over the results of Wang and Popović [1], who reported translational errors of 5–15cm along the optical axis of the tracking camera.

B. *Comparison of different matching cascades*

We evaluated the accuracy and efficiency of our system's posture estimation with regard to different matching cascades

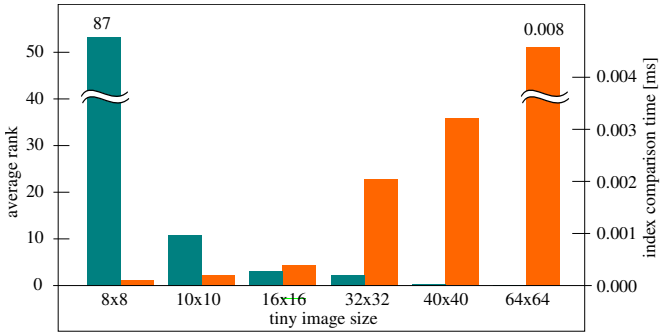

Fig. 7. Runtime efficiency and database retrieval accuracy for different tiny image sizes. Efficiency is measured by the average computation time of the distance between two images, accuracy is measured by the average rank of the true nearest neighbor in the lookup results for a database of about 8000 images over 500 trial queries. A low average rank represents high accuracy. A trade-off between efficiency and accuracy with regard to image size can be observed.

$\mathrm{DB}_{20\ \text{postures}}$

Matching Cascade	E_{posture} $[deg]$	Time $[ms]$
$16_{1000} \rightarrow 32_{300} \rightarrow 64_3$	16.61	121.6
$16_{500} \rightarrow 64_3$	16.49	91.2
64_3	16.69	178.3
8_{15}	29.68	65.1

$\mathrm{DB}_{\text{open-pinch-power}}$

Matching Cascade	E_{posture} $[deg]$	Time $[ms]$
$16_{1000} \rightarrow 32_{300} \rightarrow 64_3$	4.88	86.2
$16_{500} \rightarrow 64_3$	4.88	73.5
64_3	4.88	78.0
8_{15}	24.92	58.8

Fig. 8. Posture error and execution time for different matching cascades. The notation D_k indicates a k nearest neighbor lookup using $D \times D$ images. Arrows between two such lookups indicate the candidate pre-selection.

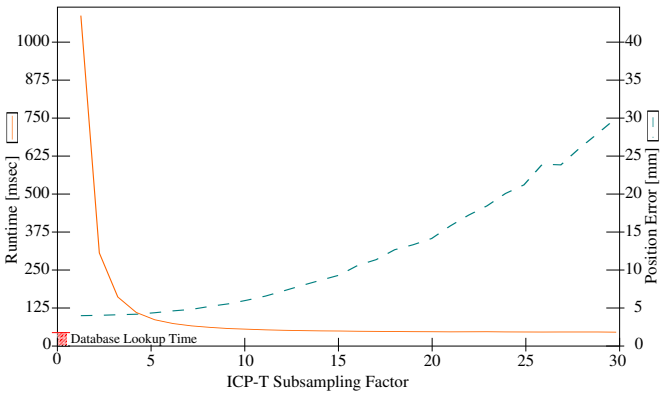

Fig. 9. Overall runtime and position estimation error in regard to an increasing point cloud sub-sampling factor.

during the database lookup. Figure 7 illustrates the accuracy/efficiency trade-off associated with different tiny image sizes. While very small images have a low distance computation cost, the information loss due to the image downscaling significantly limits their accuracy. In the database matching cascades, the smallest images can be used for a coarse pre-selection of nearest neighbor candidates in the early stages that is efficiently re-ranked using the larger images in the later stages. We performed four different matching cascades to evaluate the behavior of the overall efficiency and accuracy of our system using two example databases. Figure 8 shows the performance of the different matching cascades. The slowest database lookup uses only 64×64 images and interpolates between the 3 nearest neighbors, the fastest and least accurate one uses only 8×8 images and interpolates between the 15 nearest neighbors. Performing multiple lookup stages improves the runtime efficiency while largely maintaining accuracy. Based on these findings, we used a two-stage matching cascade in our experiments, pre-selecting 500 nearest neighbor candidates based on 16×16 images in the first stage and interpolating the 3 nearest neighbors based on 64×64 images in the second stage.

C. ICP-T sub-sampling

To improve the runtime performance we uniformly sub-sample the sensor point cloud that is used during the ICP-T pose optimization. This introduces a trade-off between pose estimation accuracy and overall efficiency. A sub-sampled point cloud is less computationally expensive to match to the virtual hand surface, but it contains less positional information. This trade-off is visualized in Figure 9, which shows that the overall runtime of our system quickly converges to the database lookup time while the position estimation error slowly rises as less points are included in the sensor point cloud. Based on this, it is possible to select a sub-sampling factor that provides a good compromise to achieve interactive framerates while maintaining high accuracy.

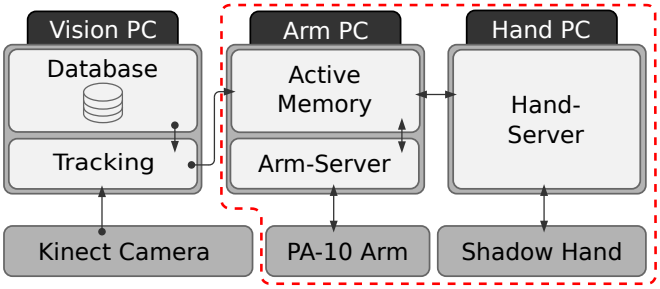

Fig. 10. System component collaboration diagram. The whole system is distributed over 3 PCs for vision, hand control and arm control. IPC is implemented using a global Active-Memory node. The dashed line highlights the robot application setup.

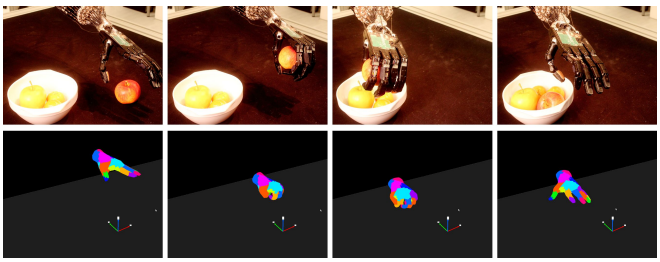

Fig. 11. Pick-and-place experiment, in which the user controls the robot hand to grasp an apple and place it in a bowl. The image sequences show the performed power grasp motion for the actuated robot hand and our hand posture and pose estimation.

V. Application

The main goal of our hand tracking system was to provide a low cost system that facilitates interactive robot control. To test this, we conducted several experiments in which we used the estimated posture and pose to actuate a compliant Shadow robot hand. This application was carried out in the Bielefeld "Curious Robot" setup [24], which has two redundant 7-DOF Mitsubishi PA-10 robot arms each equipped with a 20 DOF Shadow Dexterous Hand [25], resulting in a total of 54 DOF. The Shadow Dexterous Hands are distinguished by their human-like design: in size, number and flexibility of joints, the hands resemble their human counterparts in a very realistic manner. The entire system (see Fig. 10) is controlled by numerous processes distributed over three PCs. The processes communicate and interact using the XCF middle-ware toolkit [26], which allows for event-driven communication. We heavily use the Image Component Library (ICL) [27] for all computer-vision related tasks including camera calibration as well as 3D-visualization and rendering. The tests were carried out on a single 27-DOF PA-10/Shadow hand combination. Since the kinematic hand model used in our hand tracking system closely matches that of the Shadow robot hands (see Sec. III), the estimated joint angles can be directly transferred to the robot by sending a command to the hand-server component. The global rotation and translation of the hand is mapped from the tracker's coordinate frame to the robot's coordinate frame and transferred to the robot by issuing a command to the arm-server component. The Kinect

camera was positioned to have a top-down view of the user's hand, which provided a large working volume and minimal occlusion in our experiments.

The hand movements we investigated during the experiments ranged from general hand movements to full interaction with objects. To illustrate the effectiveness of our system we constructed a database with three different postures: *open hand*, *power grasp* and *pinch grasp*. Using these three postures, the user controlled the robot in pick-and-place tasks (see Figure 11). By observing the movements of the robot hand while performing movements as input commands, the user can utilize this direct visual feedback to naturally adjust the pose and perform the grasping motion with relative ease. We found that for well-defined tasks, such as power or pinch grasping an object, the estimation results of our system are very good with a small database containing as little as two or three postures. The speed of our system (approximately 15Hz on a PC with four Intel Xeon E5530 2.4 GHz CPU cores and 4 GBs of RAM) is completely sufficient for interactive robot actuation and the accurate 3D position estimation provided by using the Kinect camera facilitates an intuitive transfer of the user's hand motions to the robot. A live performance of our hand tracking and robot actuation system is demonstrated in the accompanying video.

VI. Discussion

Our approach to real-time hand tracking for the control of anthropomorphic robot hands is built on that of Wang and Popović [1], but differs in that we place more emphasis on retrieving an accurate hand pose than on hand posture. To produce an accurate hand posture Wang and Popović implemented fast similarity sensitive coding [20], allowing more postures to be present in the database while maintaining near real-time performance. They also added a 2D inverse kinematics step which adapts the estimated hand posture to that of the current camera image. These features mean that in terms of the estimated posture, our system is not as robust as that of Wang and Popović's. However, for the task of controlling a compliant robot hand in tasks such as grasping, an accurate pose is much more important than an exact approximation of the user's hand posture. In our approach we achieve a higher accuracy in pose estimation by using a Kinect camera and color sensitive ICP-T. We argue that database approaches such as ours can be used to provide a good initial estimate of the posture and pose of the hand and then this can be used as a pre-initialization step for model-based tracking approaches that suffer from situations in which the hand is temporarily lost.

We have observed that under certain conditions, such when there are occlusions or the database is sparsely populated, it is possible that the output is an interpolation between two quite distinct postures in the database. While this is a feature of our system, allowing us to compute smooth transitions between discrete entries in the database, it can result in large discrepancies between the real posture and that given by the system. An inverse kinematics step could improve on this, however, as our focus was on the control of an anthropomorphic robot, accurate estimation of the pose of the hand took precedence over the posture. Using a combination of visual feedback, allowing us to adapt our hand during robot control according to what the situation required, and taking advantage of the compliance in the robot hand, we were able to smoothly actuate the robot hand to perform various grasping and interaction tasks.

In future work there are a number of avenues that we are considering to improve our current system. Efficiency could be improved by moving from a pure CPU implementation, as is done now, to an implementation on the GPU, especially for the database lookup step involving image matching. For smaller databases of postures the bottleneck is still the color sensitive ICP-T algorithm and this could be optimized using a k-d tree for finding the closest triangle to a Kinect sample point. Adding depth values from the Kinect camera to the database itself has the potential to considerably improve on the estimated posture and indeed could be a step towards not needing a color glove at all.

ACKNOWLEDGMENT

The authors are grateful to Robert Wang for his valuable feedback and for providing the design of the color glove. This work was supported by the DFG Center of Excellence "Cognitive Interaction Technology" (CoE 277: CITEC) and the DFG grant "Real-Time Acquisition and Dynamic Modeling of Human Faces, Upper Bodies, and Hands" (BO 3562/1-1).

References

[1] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 63:1–63:8, 2009.

[2] "Microsoft Corp. Redmond WA. Kinect for Xbox 360."

[3] A. M. R. Agur and M. J. Lee, *Grant's Atlas of Anatomy*, 10th ed. Lippincott Williams and Wilkins, 1999.

[4] J. Maycock, J. Steffen, R. Haschke, and H. Ritter, "Robust tracking of human hand postures for robot teaching," in *IEEE/RSJ Int. Conf. on Intel. Robots and Systems (IROS)*, 2011, pp. 2947–2952.

[5] Y.-H. Lee and C.-Y. Tsai, "Taiwan sign language (tsl) recognition based on 3D data and neural networks," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1123–1128, 2009.

[6] M. Fischer, P. van der Smagt, and G. Hirzinger, "Learning techniques in a dataglove based telemanipulation system for the DLR hand," in *Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 1998, pp. 1603–1608.

[7] W. Griffin, R. Findley, M. Turner, and M. Cutkosky, "Calibration and mapping of a human hand for dexterous telemanipulation," in *Proc. ASME Int. Mechanical Engineering Congress & Exposition (IMECE), "Haptic Interfaces for Virtual Environments and Teleoperator Systems" Symposium*, 2000, pp. 1145–1152.

[8] M. Turner, "Programming dexterous manipulation by demonstration," PhD Thesis, Stanford University, Department of Mechanical Engineering, Stanford, USA, 2001.

[9] J. Steffen, J. Maycock, and H. Ritter, "Robust dataglove mapping for recording human hand postures," in *International Conference on Intelligent Robotics and Applications (ICIRA)*. Springer, 2011, pp. 34–45.

[10] H. Hamer, K. Schindler, E. Koller-Meier, and L. V. Gool, "Tracking a hand manipulating an object," in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 1475–1482.

[11] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints," in *13th IEEE Int. Conf. on Computer Vision (ICCV 2011)*, Barcelona, Spain, 2011, pp. 2088–2095.

[12] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," in *22nd British Machine Vision Conference ($BMVC$ 2011)*, University of Dundee, UK, 2011.

[13] T. Lozano-Perez, G. Gallagher, L. P. Kaelbling, and R. Tedrake, "Kinect Hand Detection," http://www.csail.mit.edu/videoarchive/research/hci/kinect-detection, 2010, Accessed: Jul 2012.

[14] R. Wang, S. Paris, and J. Popović, "6d hands: markerless hand-tracking for computer aided design," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2011, pp. 549–558.

[15] M. Do, J. Romero, H. Kjellstrm, P. Azad, T. Asfour, D. Kragic, and R. Dillmann, "Grasp recognition and mapping on humanoid robots," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2009.

[16] J. Romero, H. Kjellstrm, and D. Kragic, "Hands in action: Real-time 3d reconstruction of hands in interaction with objects," in *In: IEEE International Conference on Robotics and Automation*, 2010.

[17] J. Maycock, D. Dornbusch, C. Elbrechter, R. Haschke, T. Schack, and H. Ritter, "Approaching manual intelligence," *Künstliche Intelligenz – Issue Cognition for Technical Systems*, pp. 287–294, 2010.

[18] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[19] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: two new techniques for image matching," in *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., 1977, pp. 659–663.

[20] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, Jun. 2008.

[21] D. Jacka, A. Reid, and B. Merry, "A comparison of linear skinning techniques for character animation," in *In Afrigraph*. ACM, 2007, pp. 177–186.

[22] P. J. Schneider and D. Eberly, *Geometric Tools for Computer Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[23] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[24] I. Lütkebohle, J. Peltason, L. Schillingmann, C. Elbrechter, B. Wrede, S. Wachsmuth, and R. Haschke, "The Curious Robot – Structuring Interactive Robot Learning," in *Int. Conf. on Robotics and Automation (ICRA)*, Kobe, 2009.

[25] Shadow Robot Company, "The Shadow Dextrous Hand." [Online]. Available: http://www.shadowrobot.com/hand/overview.shtml

[26] C. Bauckhage, S. Wachsmuth, M. Hanheide, S. Wrede, G. Sagerer, G. Heidemann, and H. Ritter, "The visual active memory perspective on integrated recognition systems," *Image and Vision Computing*, vol. 26, no. 1, pp. 5–14, 2008.

[27] C. Elbrechter, M. Götting, and R. Haschke, "Image Component Library (ICL)," Jan 2010, http://iclcv.org.