GraspDB14 – Documentation on a database of grasp motions and its creation

Katharina Stollenwerk
André Hinkenjann
Reinhard Klein

Publisher: Dean Prof. Dr. Wolfgang Heiden

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science

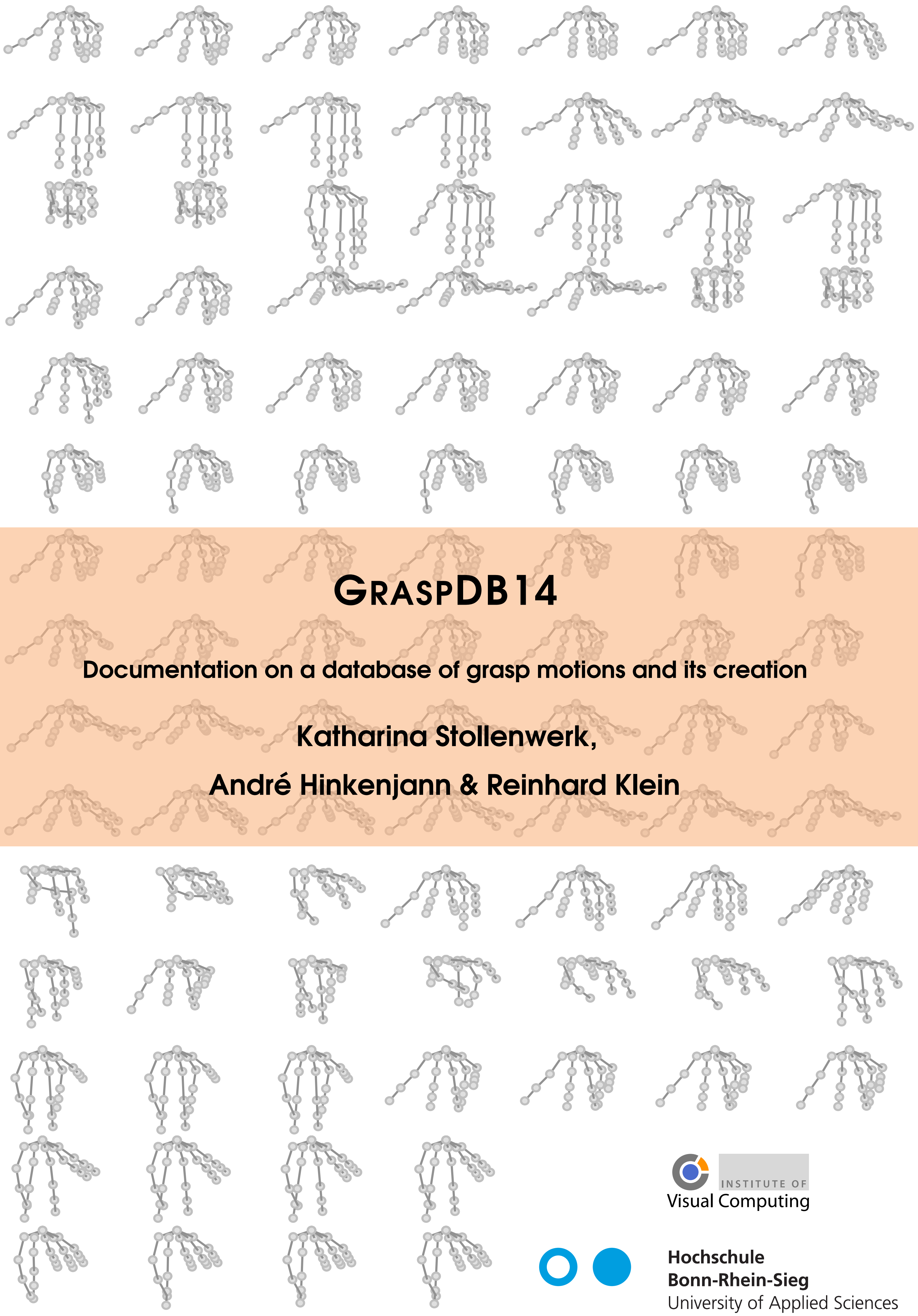
Sankt Augustin, Germany

January 2018

Technical Report 01-2018

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272 ISBN 978-3-96043-061-2



GraspDB14

Documentation on a database of grasp motions and its creation

Katharina Stollenwerk,

André Hinkenjann & Reinhard Klein

Institute of Visual Computing

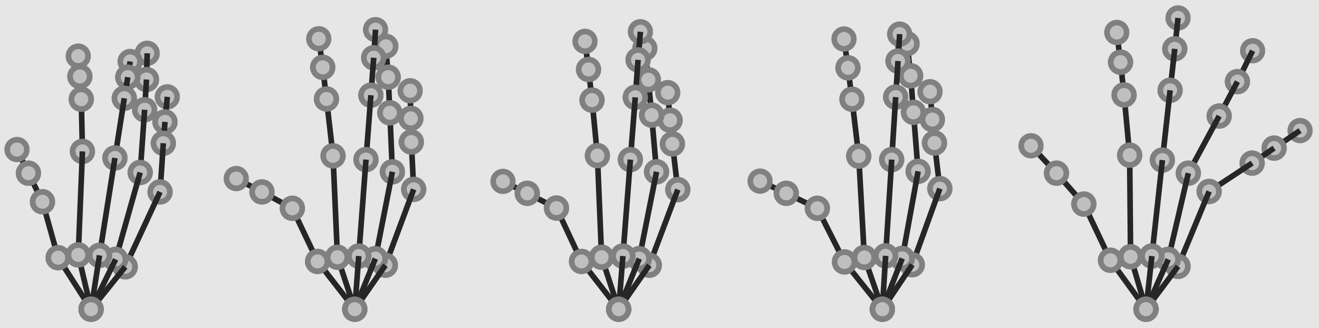
Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

PUBLISHED BY DEAN PROF. DR. WOLFGANG HEIDEN, DEPARTMENT OF COMPUTER SCIENCE

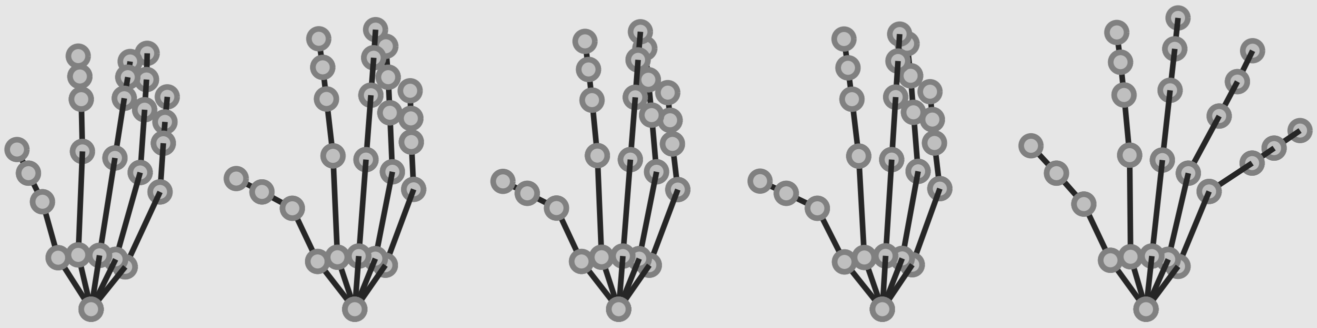
TECHNICAL REPORT, NO. 01-2018, HOCHSCHULE BONN-RHEIN-SIEG, JANUARY 2018
ISSN 1869-5272
ISBN 978-3-96043-061-2

DATABASE: HTTPS://SKYLAB.VC.H-BRS.DE/KSTOLL2M/GRASPDB14
DOI: HTTPS://DOI.ORG/10.18418/978-3-96043-061-2

First printing, January 2018



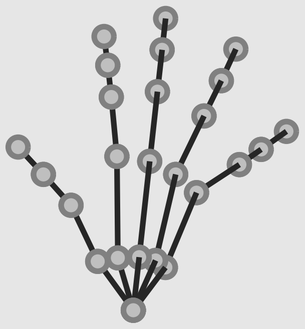
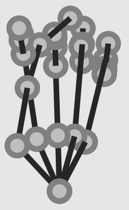
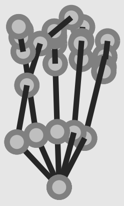
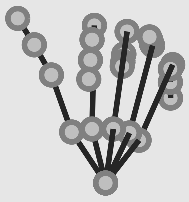
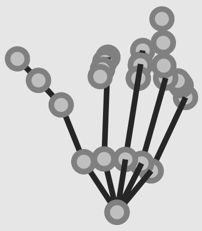
Contents



List of tables and list of figures

List of tables

List of figures

1. Preface

Motion capture, often abbreviated mocap, generally aims at recording any kind of motion – be it from a person or an object – and to transform it to a computer-readable format. Especially the data recorded from (professional and non-professional) human actors are typically used for analysis in e.g. medicine, sport sciences, or biomechanics for evaluation of human motion across various factors. Motion capture is also widely used in the entertainment industry: In video games and films realistic motion sequences and animations are generated through data-driven motion synthesis based on recorded motion (capture) data.

Although the amount of publicly available full-body-motion capture data is growing, the research community still lacks a comparable corpus of specialty motion data such as, e.g. prehensile movements for everyday actions. On the one hand, such data can be used to enrich (hand-over animation) full-body motion capture data - usually captured without hand motion data due to the drastic dimensional difference in articulation detail. On the other hand, it provides means to classify and analyse prehensile movements with or without respect to the concrete object manipulated and to transfer the acquired knowledge to other fields of research (e.g. from 'pure' motion analysis to robotics or biomechanics). Therefore, the objective of this motion capture database is to provide well-documented, free motion capture data for research purposes.

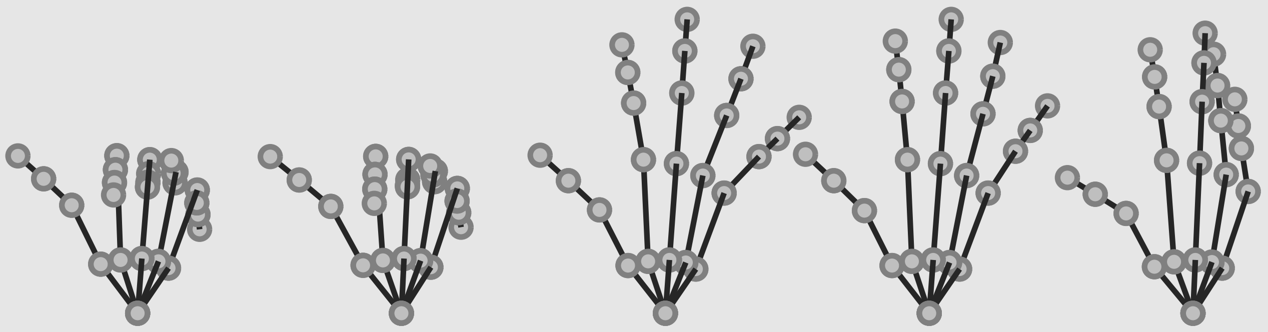
The presented database GRASPDB14 in sum contains over 2000 prehensile movements of ten different non-professional actors interacting with 15 different objects. Each grasp was realised five times by each actor. The motions are systematically named containing an (anonymous) identifier for each actor as well as one for the object grasped or interacted with. The data were recorded as joint angles (and raw 8-bit sensor data) which can be transformed into positional 3D data (3D trajectories of each joint).

In this document, we provide a detailed description on the GRASPDB14-database as well as on its creation (for reproducibility). Chapter 2 gives a brief overview of motion capture techniques, freely available motion capture databases for both, full body motions and hand motions, and a short section on how such data is made useful and re-used. Chapter 3 describes the database recording process and details the recording setup and the recorded scenarios. It includes a list of objects and performed types of interaction. Chapter 4 covers used file formats, contents, and naming patterns. We provide various tools for parsing, conversion, and visualisation of the recorded motion sequences and document their usage in chapter 5.

GraspDB14 as well as the accompanying C++ and MATLAB tools are publicly available under `https://skylab.vc.h-brs.de/kstoll2m/GraspDB14`.

Any comments and suggestions for improvements are appreciated.

Katharina Stollenwerk



2. Related work and motivation

2.1 Overview of motion capture techniques

Motion capture comes in many different flavours, mainly classifiable into the broad field of optical system and the narrower field of non-optical systems. *Optical systems* typically make use of two or more image sensors to capture the scene and compute the 3D position(s) of a target by triangulation. Optical system can be further categorised into *marker-based systems*, e.g. [Opt17; Vic17], and *markerless systems*, e.g. [Lea17; Sho+13]. Marker-based systems make use of active markers emitting light or of passive markers. Passive markers usually are retro-reflective reflecting light emitted from a light source close to the capturing camera. In markerless systems, depth cameras or "regular" RGB or IR cameras are used in combination with computer vision algorithms capable of identifying human forms and decomposing them in parts suitable for tracking.

Non-optical systems include inertial systems (using miniature inertial sensor in combination with predefined skeleton models and sensor fusion algorithms [RLS13]) and mechanical motion capture (direct measuring of body joint angles using articulate mechanical parts (exoskeleton, e.g. [Met17]) or bend sensors (data glove for hands, e.g. [Cyb17]). For a general overview of motion capture technologies and techniques especially tailored to the acquisition of hand motion data refer to the extensive state-of-the-art report of Wheatland et al. [Whe+15].

2.2 Publicly available motion capture databases

While there are a number of high-quality, full-body motion capture databases that can be used for academic purposes (e.g. the CMU and HDM05 motion capture databases [CMU13; Mül+07]), only a few such data collections exist for articulated hand motions.

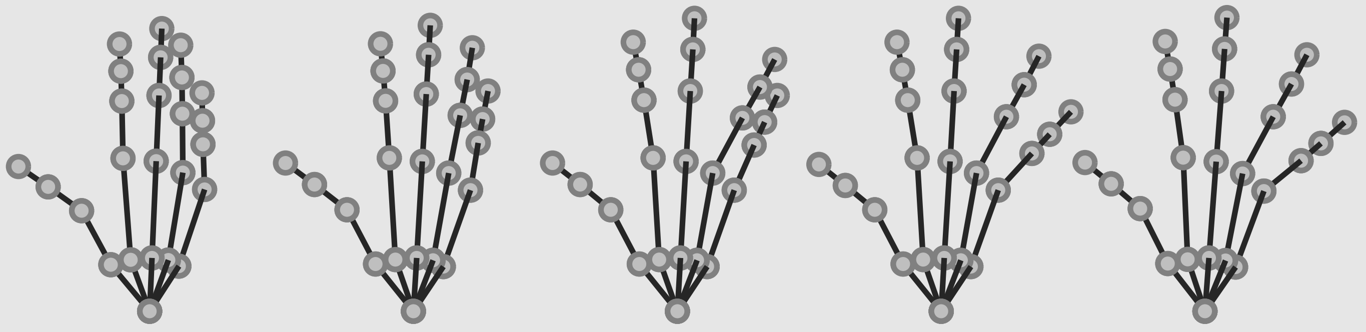
In the field of robotics, Goldfeder et al. [Gol+09] presented algorithms for automatic generation of a database of precomputed stable grasps, i.e. a single pose, for robotic grasping. This resulted in *The Columbia grasp database* which contains computed grasp configurations of different (robotic) hands along with a set of graspable objects. Thus, the database only contains single-hand poses and no finger movements. Feix et al. [Fei+13] provide a small dataset of human grasping (Human Grasping Database) used as a basis for evaluation of the motion capabilities of artificial hands. The dataset contains 31 motions, each performed by five subjects twice. The motion data contains only the 3D position and orientation of each fingertip with no information on the specific underlying hand model. Notable also due to its size, the *NinaPro database* [Atz+12] contains 52 full-hand and wrist motions ranging from hand postures over functional movements to prehensile movements. They recorded a total of 52 full hand and wrist motions, therein 23 grasping and functional movements, of 27 subjects. Each execution of a motion was restricted to a five second time frame in which the subjects had to mimic a predefined grasp in sync with a displayed video of the prehensile movement. Each motion was executed ten times. The data recorded consists of surface electromyography (sEMG) data together with the 8-bit valued raw output of a 22-sensor CyberGlove (kinematic data). The database was released for the biorobotics community aiming at providing a benchmark for testing and developing machine learning algorithms for hand-movement-sEMG-data widely used for myoelectric hand prosthesis control methods. The database is available for academic purposes but the 8-bit valued raw kinematic recordings of each sensor only roughly represent joint angles and fail to account for cross coupled sensors in the CyberGlove. While this poses no big issue when learning hand movement sEMG data it does when aiming at re-using the data for animation.

2.3 Re-use of motion capture data

A vital subtask in the re-use of recorded motion capture data is the ability to split motion clips or motion sequences into primitive parts (segments) which can be used for motion analysis and synthesis. As manual segmentation and annotation of motion data into meaningful phases is a tedious and daunting task it is essential that this is automated. The following paragraphs outline some of the works on temporal segmentation of motion capture data, including what the segmentation was ultimately used for.

For full-body motion data, unsupervised, temporal segmentation techniques have been developed. Among them is the work of Beaudoin et al.'s [Bea+08] who focus on visualising the structure of motion datasets. They propose to partition motion data streams into motion-motifs by detecting and clustering temporally repeating motion patterns. These motion-motifs are organised in a graph structure useful for motion blending and motion data compression. Zhou et al. [ZTH08; ZTH13] segment human motion data based on (hierarchical) aligned cluster analysis (H(ACA)). They frame the task of motion segmentation as temporal clustering of (motion capture) data into classes of semantically-similar motion primitives. Min and Chai's Motion Graph++ [MC12] is not only capable of segmenting motions into basic motion units, but also of motion recognition and synthesis. Their segmentation automatically extracts keyframes of contact transitions and semi-automatically extracts keyframes exhibiting salient visual content changes. Vögele et al. [VKK14] employ (backward and forward) region growing in order to identify start and end frames of activities (groups of motion primitives). These activities are split into motion primitives by taking advantage of the observation that repetitive patterns in activities manifest in minor diagonals in a respective sparse self-similarity matrix (SSSM).

In the field of robot programming and teaching from example, e.g. in pick and place scenarios, researchers have looked into temporal segmentation of recorded human hand and finger motions. This is often needed to characterise grasp phases (e.g. pre-grasp, grasp, manipulation) [EK05; KI94]. Other works related to temporal segmentation of captured hand motion data mainly focus on segmenting conversational hand gestures into different phases for synthesis. Often this segmentation is based on an accompanying audio stream [JHS12; LTK09; MAN15]. Based on the previously mentioned concept of SSSMs, Stollenwerk et al. [Sto+16] segmented articulated hand motion data (prehensile movements) into meaningful phases. They also successfully apply this segmentation approach for classification of grasp motions.



3. Recording of the database

3.1 Contributors

The database was designed and recorded by Katharina Stollenwerk, Hochschule Bonn-Rhein-Sieg, Germany. The CyberGlove was kindly provided by Matthias Bues, Fraunhofer IAO, Stuttgart, Germany. The following people contributed to the creation of this database (in alphabetical order): Matthias Bues (equipment), André Hinkenjann (expertise), Reinhard Klein (expertise), Katharina Stollenwerk (design, execution, acquisition, scripts, protocols, data cleaning). Family, colleagues, and students from Hochschule Bonn-Rhein-Sieg and its Institute of Visual Computing, Sankt Augustin, Germany, supported the creation of this database by patiently performing the given tasks.

Please send comments and suggestions for improvements to

Katharina Stollenwerk (kstollen.cs@gmail.com).

3.2 Recording setup

This section provides information on the recording procedure. In addition, details on the equipment used are given and the setups for recording and measuring are outlined.

3.2.1 Procedure - The same procedure as every year, James.

The recording of the motions was based on a fixed script ensuring inter-actor consistency. The general steps will be laid out here and detailed in the following sections 3.3 through 3.6. The general script consists of four parts the last of which is further subdivided into three scenarios. After the recording of each scenario there was a break.

Part 1 - General descriptions (equipment, tasks, scenarios), see section 3.3.
Part 2 - Measuring of the actor's hand, see section 3.4.
Part 3 - Recording of the CyberGlove calibration sequence, see section 3.5.
Part 4 - Recording of the dababase scenarios, see section 3.6.
  Scenario 1 - Object transport (uncontrolled), see section 3.6.1.
  Scenario 2 - Object use and interaction, see section 3.6.2.
  Scenario 3 - Object transport (controlled), see section 3.6.3.

3.2.2 Equipment, setups, and technology involved

For measuring general properties of the actor's hand an off-the-shelf tape measure was used. Additionally, a calibrate image of the hand was taken in order to later be able to measure certain more specific landmarks of the hand (section 3.4). These images were taken with a Canon EOS 30D and a Canon 28-135 mm ultrasonic lens. A symmetric dot grid with an inter-dot-distance (centre-to-centre) of 5 cm and a dot diameter of 1 cm was used as calibration target as depicted in figure 3.3a. The camera was located vertically above the calibration target.

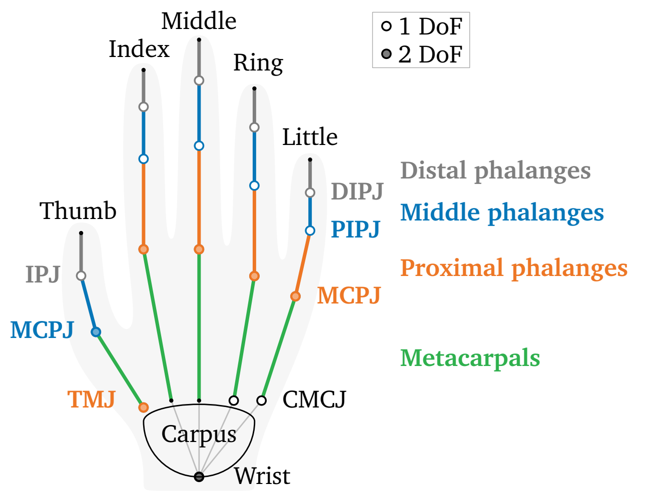
Motion data was recorded using a Virtual Technologies 18-sensored CyberGlove [Vir98]. The CyberGlove is a dataglove with 18 bend sensors monitoring motions in the hand's joints (flexion and extension) and between fingers (adduction and abduction). The 18-sensored model does not include sensors over the distal interphalangal joints. As a consequence, the fingertips of the glove are left open. In their manual, they argue that the distal interphalangeal joint can, in most cases, be inferred from the other joints in that finger [Vir98]. They further point out the advantages of the open fingertips in the glove, which makes it more comfortable to wear and preserves tactile sensitivity. Figure 3.1b depicts the position of the CyberGlove's sensors with respect to a *model skeletal kinematic chain* as shown in figure 3.1a. Table 3.1 lists sensor denotations used and a general description of what each sensor monitors. The sensor numbers (the colmun labelled 'No.') in the table match the sensor numbers in the figure.

Communication with the CyberGlove was implemented through the *Virtual-Reality Peripheral Network (VRPN)* [ST17; Tay+01]. For this purpose, a small server client module was written in C++ which sends commands to the glove and retrieves data from the glove.

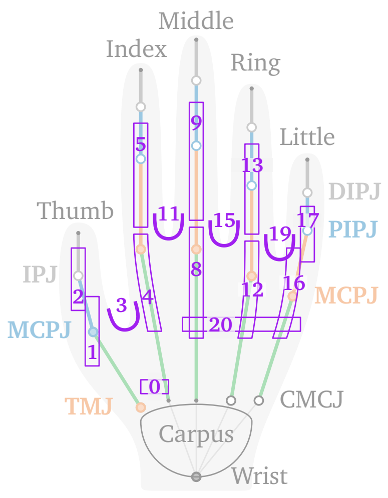
For the recording of the individual scenarios the actors sat at a table. The objects to be manipulated were placed on the table one after the other. The actors then either interacted with the object or grasped it in order to lift it up and place it somewhere else on the table, close to where the object was originally put.

No.	Sensor	Description
0	T-TMJ	angle of thumb rotating across palm at trapeziometacarpal joint
1	T-MCPJ	flexion/extension at thumb metacarpophalangeal joint
2	T-IPJ	flexion/extension at thumb interphalangeal joint
3	TI-AA	angle between thumb and index finger
4	I-MCPJ	flexion/extension at index finger MCP joint
5	I-PIPJ	flexion/extension at index finger proximal interphalangeal joint
8	M-MCPJ	flexion/extension at middle finger MCP joint
9	M-PIPJ	flexion/extension at middle finger PIP joint
11	MI-AA	angle between middle and index fingers
12	R-MCPJ	flexion/extension at ring finger MCP joint
13	R-PIPJ	flexion/extension at ring finger PIP joint
15	RM-AA	angle between ring and middle fingers
16	L-MCPJ	flexion/extension at little finger MCP joint
17	L-PIPJ	flexion/extension at little finger PIP joint
19	LR-AA	angle between little and ring finger
20	P-Arch	angle of little finger rotating across palm

Table 3.1: CyberGlove sensor denotations along with general descriptions. The numbers ('No.') are in accordance with the sensor labels in figure 3.1b. In the sensor denotations, AA abbreviates abduction/adduction. The information has been partially taken from [Vir98].



(a) Skeletal kinematic chain model of the hand.



(b) Positions of the CyberGlove sensors.

Figure 3.1: In the skeletal kinematic chain model of the hand (a) rigid bones (represented by lines) are connected by joints (circles) of different degrees of freedom (DoF). Filled circles represent joints with two degrees of freedom (DoF) in that joint, unfilled circles represent joints with one DoF. Abbreviated joints spell out [proximal, distal] interphalangeal joints ([P,D]IPJ), metacarpophalangeal joint (MCPJ), carpometacarpal joint (CMCJ), and trapeziometacarpal joint (TMJ). (b) shows the positions of the CyberGlove sensors with respect to the skeletal kinematic chain in (a). Sensors are marked by violet rectangles (joint sensors) or curves ([ad/ad]duction sensors). Each sensor is labelled with its sensor ID. Details on the sensor IDs are listed in table 3.1.

3.3 General description (equipment, tasks, scenarios)

Before recording began, the actors were familiarised with the equipment used (CyberGlove) and the general tasks they would have to perform (grasping and interacting with objects) in order to give them a overall overview. It was also explained how the individual scenarios would be recorded.

3.4 Measuring the actor’s hand

Measuring the hand is two-step procedure: The first step consists of taking measurements at certain general characteristics of the hand with a tape measure. The features measured are

- the palm circumference at the metacarpophalangeal joints (knuckles),
- the wrist circumference, and
- the length of the vertical axis of the hand (distance of the tip of the middle finger to the wrist).

Figure 3.2 illustrates where measurements were taken and table 3.2 lists the resulting lengths for all recorded actors.

Aside from tape measurements of general characteristics of the hand, other more specific landmarks such as positions of joints were recorded. For that purpose, the joint positions were marked with a felt-tip pen on the skin of the actor’s back of the hand, see figure 3.3a. After camera calibration and removal of the calibration target, the actor placed their hand into the calibrated region and an image of the hand with the marked joint positions was taken. That way, the lengths of the hand as well of specific bones can later be measured in the image, compare figures 5.1(b) and 5.2. The camera calibration file as well as the image of the actor’s hand are saved.

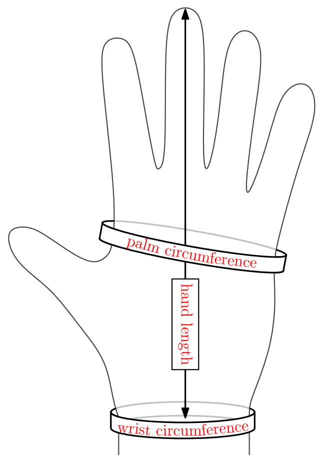
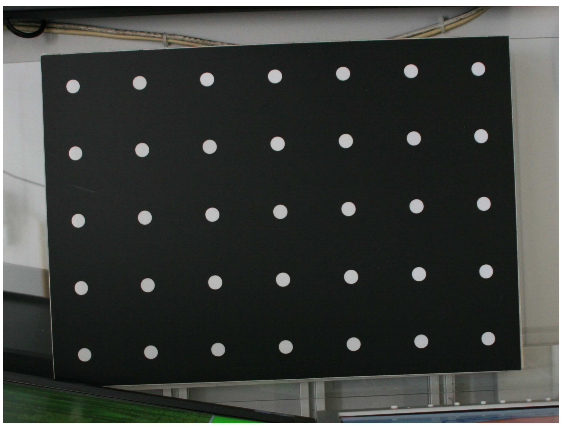


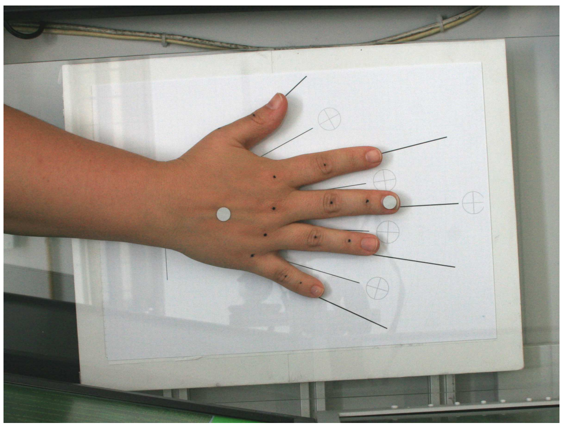
Figure 3.2: Locations for taking hand measurements.

Actor ID	Palm circumference	Hand length	Wrist circumference	m/f
1	19.5 cm	18.0 cm	16.5 cm	f
2	22.5 cm	20.0 cm	18.5 cm	m
3	18.3 cm	18.5 cm	15.5 cm	f
4	19.5 cm	17.8 cm	17.0 cm	f
5	22.0 cm	20.5 cm	19.5 cm	m
6	22.0 cm	19.6 cm	19.0 cm	m
7	22.0 cm	20.5 cm	18.5 cm	m
8	19.5 cm	20.0 cm	18.0 cm	m
9	22.5 cm	19.5 cm	17.5 cm	m
10	21.3 cm	19.5 cm	17.8 cm	m

Table 3.2: Hand measurements of all actors. The positions in which measurements were taken are shown in figure 3.2.

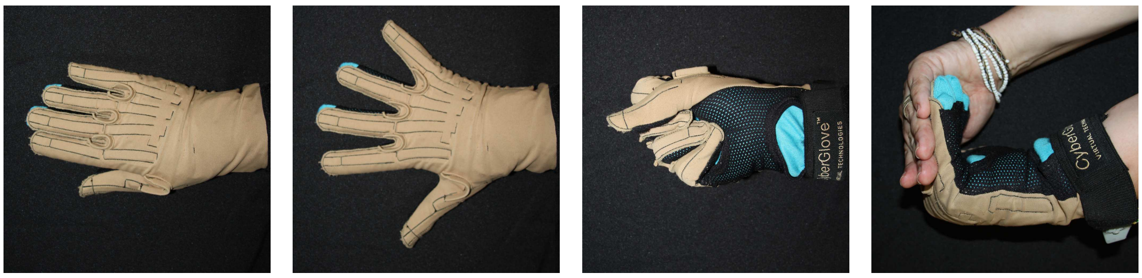


(a) Calibration target.



(b) An actor's hand with felt-tip marked joints.

Figure 3.3: Calibration process and hand marked with a felt-tip pen. (a) depicts the calibration target used for camera calibration and (b) shows the position of and the drawn felt-tip drawn dots on an actor's hand. The two reflective markers on the centre of the back of the hand and the middle finger tip remained unused throughout measuring and recording.



(a) Flat unspread hand (b) Flat spread hand (c) Fist, thumb outside (d) Square angle in MCPJ

Figure 3.4: Static poses used in the recording of the calibration sequence. In (a) the angles between the fingers should be (close to) zero while in (b) the fingers should be spread as far apart as possible. For (d) the ungloved hand is used to stabilise the fingers of the hand wearing the glove.

3.5 Recording of the CyberGlove calibration sequence

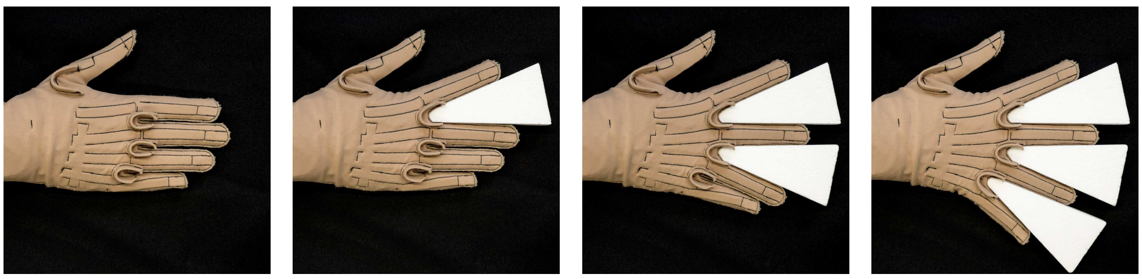
In order to provide a possibility to calibrate the sensors of the CyberGlove, we recorded a calibration sequence for each actor. The sequence consists of several static poses as well as short motions. The calibration sequence is recorded in one continuous take. Switching of poses or between motions during the take are marked using the CyberGlove's switch state (ON/OFF) which is controlled by a physical button on the glove itself. At the beginning of the recording, the switch's sate is ON.The calibration sequence consists of the following four parts:

Static poses

The actor is asked to mimic the four static poses depicted in figure 3.4. As soon as the pose is assumed the glove switch button is pressed twice with a short delay between pressing. Hence it is turned to OFF for a short period of time during which the pose is held. The poses are described as 'flat unspread hand' (figure 3.4a), 'flat spread hand' (figure 3.4c), 'fist, thumb outside' (figure 3.4c), and 'square angle in MCPJ' (figure 3.4d). Each pose is illustrated with the corresponding image.

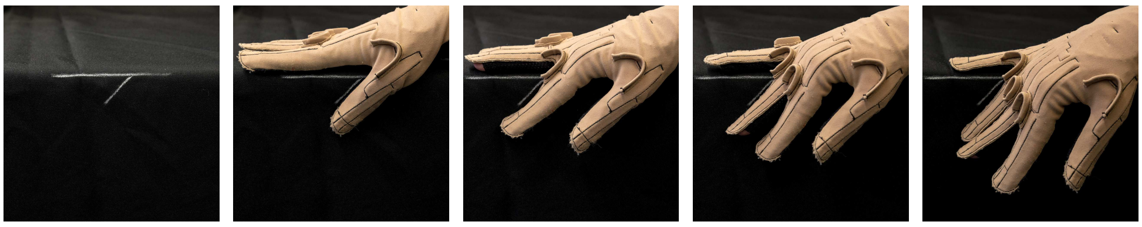
Finger ad-/abduction by 25 degrees

Here, the actor's hand lies flat on a table. Three polystyrene wedges of 25 degree with a rounded tip are placed one after the other between index and middle finger, between middle and ring finger, and between ring and little finger (see image sequence in figure 3.5). The CyberGloves switch is pressed after each wedge is in place. So, after finishing these three poses, the final switch state is OFF. Afterwards, the wedges are removed in the same succession as they were put in, and fingers are moved back together. Again, the switch is pressed after completion of each motion (the final switch state after this part is ON).



(a) No wedges (b) First wedge in place (c) Two wedges in place (d) Three wedges in place

Figure 3.5: Finger adduction/abduction by 25 degrees calibration sequence. The image succession represents the succession of placing the wedges between pairs of fingers one after the other.



(a) Lines drawn for orientation (b) Thumb (c) Index finger (d) Middle finger (e) Ring finger

Figure 3.6: Finger flexion poses used in the recording of the calibration sequence. In order to align the finger pairs with the drawn line at 45 degrees the hand was moved horizontally.

Finger flexion in the metacarpophalangeal joints by 45 degrees

For this part, the vertical edge of a table had been marked with a line at an angle of 45 degrees. The actor placed the hand onto the table such that the finger to flex hovered in the air and horizontally aligned with the edge of the table. The finger was then flexed in the metacarpophalangeal joint until the line on the table border was just above the upper side of the finger. Angles were applied between thumb and index finger, between index and middle finger, between middle and ring finger, and between ring and little finger in that succession as depicted in figure 3.6. After each finger pair was in position, the glove's switch was pressed, leaving it in the ON-state after this part.

Thumb to finger touch

This part consists of two motion sequences. In the first, the actor performs a finger tip touch between thumb and index finger, thumb and middle finger, thumb and ring finger, and thumb and little finger. The CyberGlove switch is pressed after the last finger tip touch (switch state then: OFF). In the second sequence, the thumb tip is to touch each finger's middle phalanx.

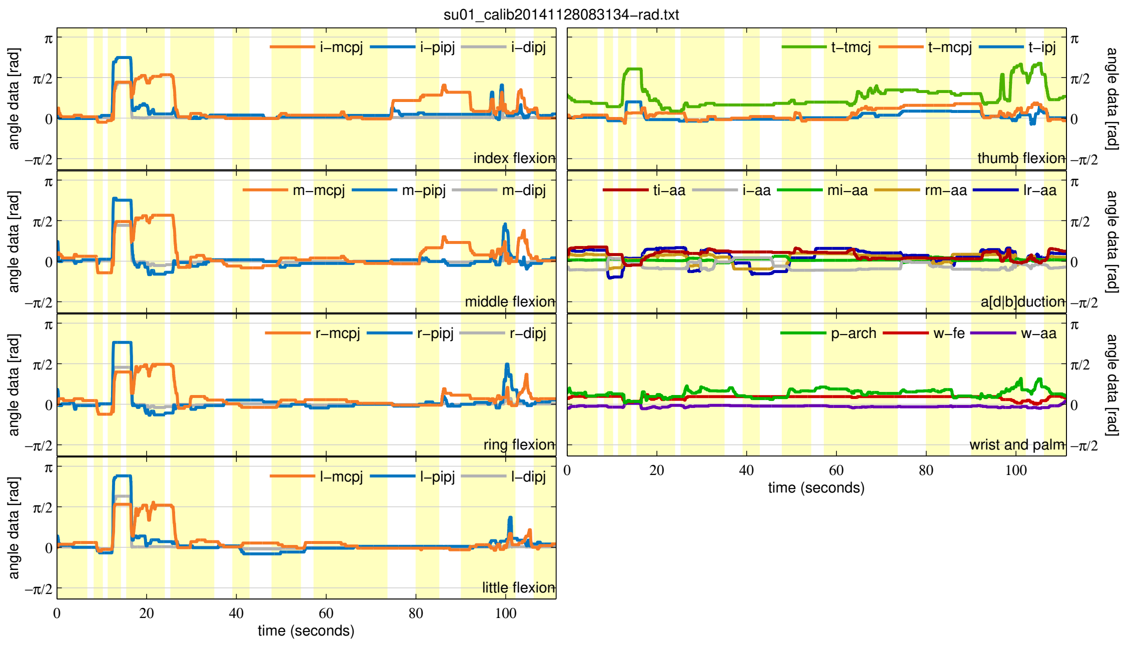


Figure 3.7: Exemplary angle trajectories recorded in a glove calibration sequence. Yellow indicates that the switch state is ON. Its state is OFF in the white portions of the plots. For better readability, flexion/extension-trajectories are grouped by finger and ad-/abduction trajectories are plotted together. Each finger is represented by its first letter, 'w' stands short for 'wrist' and 'p' for palm.

Data recording is stopped after the thumb has touched the little finger's middle phalanx. Some participants pressed the switch again and the recording was stopped a little later. As a result the calibration sequences consist of a total of 20 (or 21) ON/OFF-segments. A sample calibration recording with 21 ON/OFF-segments is drawn in figure 3.7.

3.6 Recording of the database scenarios

To the best of our knowledge there still is no database of articulated human hand motions covering a wide variety of actions and actors usable for motion analysis and data-driven synthesis. We therefore decided to create one.

We chose to include three scenarios in the database: *object transport (uncontrolled)*, *object use and interaction* and *object transport (controlled)*, all of which will be described below. To ensure inter-person consistency in the recording setups and for later reproducibility, a protocol was written detailing each step of the recording.

In all scenarios the hand was placed flat on a table before and after task execution. During the executing of each task, the CyberGlove's switch state is enabled (i.e. set to ON): The actors enabled the state before and disabled it after each task execution. Consequently, each task has a predefined succession of actions, so that the complete motion sequence can later be easily separated into individual tasks (also called *motion trials*), see section 3.7. The data were recorded at an acquisition rate of 60[Hz] using an 18-sensor right-hand Virtual Technologies CyberGlove as depicted in figure 3.4. Each hand pose (a set of 18 data points per sample) is represented by a joint angle configuration of the joints in each finger, wrist and palm.

3.6.1 Object transport (uncontrolled)

In *uncontrolled object transport*, each actor is presented with a variety of objects in random order with no object being presented twice in a row. The task is to pick up the object, move it to a different location, and put it down. The scenario was designed in order to obtain a high diversity of possible grasps per object. This scenario comprised 14 different objects one of which was presented in two different orientations (the carton was presented standing and lying flat). Each object had to be lifted and moved five times. Figure 3.11 gives an overview of all objects used for this scenario. Additionally, figures A.1 through A.3 depict the objects individually also showing them from different perspectives.

3.6.2 Object use and interaction

In *object use and interaction*, the actor is again presented with a variety of objects but in a specific order. The task depended on the object presented. It covered opening and closing of objects with the left and right hand. Tightly closing (screwing) objects and using objects by their design (e.g. writing with a pen, lifting a cup of glass and drinking from it, throwing a ball, or pouring water from a bottle into a mug). In detail, the general tasks in this scenario were the following:

Right handed opening/closing

First task: open the object using the right hand (i.e. the lid or cap should be in the right hand) and put the lid (or cap) and the object onto the table. Second task: close the object using the right hand and put it onto the table. Objects used were (in that succession): a bottle, an oval jar, a pen and a classic notebook.

Left handed opening/closing

The tasks were identical to right handed opening/closing switching hands. I.e. the object was to be held with the right hand and the action was performed with the left hand.

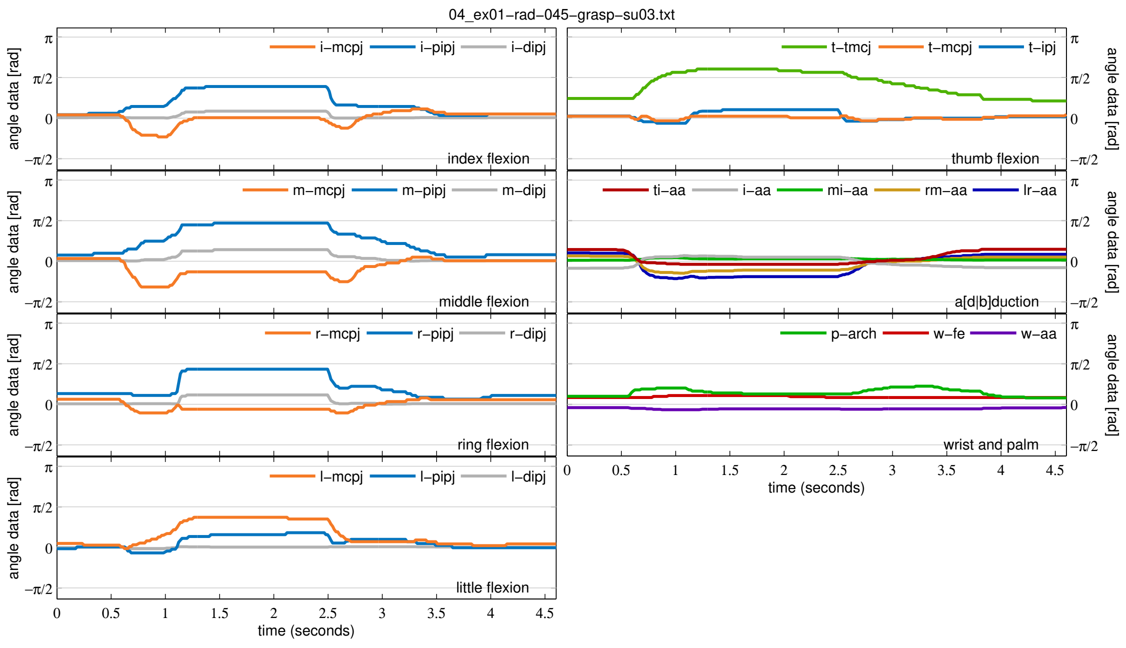


Figure 3.8: Exemplary angle trajectories recorded in *uncontrolled object transport*. The object lifted and moved was a mug. Angle trajectories are grouped by finger and type of movement (flexion/extension, adduction/abduction).

Right handed tight closing

Here, the task was to tightly screw the lid of the object onto the object using the right hand (i.e. the lid is in the right hand) and to put the object back onto the table. Objects used in this part were a bottle and an oval jar.

Left handed tight closing

Repeat the right handed task above with the left hand.

Object use

The main task for this part of the scenario was to 'naturally' interact with the object presented. The objects and tasks were (in the succession given):

Pen Take the pen, remove the cap, write "The quick brown fox jumps over the lazy dog", close the pen, and put it aside.

Bottle Take the bottle, pour liquid into the mug standing there, and put the bottle back down. The bottle is filled with water but stays closed during the entire task as it may slip when taking it into the gloved hand.

Mug Take the mug and drink from it. Put the mug back onto the table.

Glass Take the glass and drink from it. Put the glass back down.

Ball Take the ball and throw it.

3.6.3 Object transport (controlled)

In *controlled object transport*, each actor is again presented with a variety of objects, this time in a fixed order. The task again is to pick up the object, move it to a different location, and put it down. Only here, a picture illustrates how to hold the object during transport. The task has to be executed five consecutive times on each object. Contrary to the first scenario, the focus here lied on reproducing consistent, predefined grasp motions.

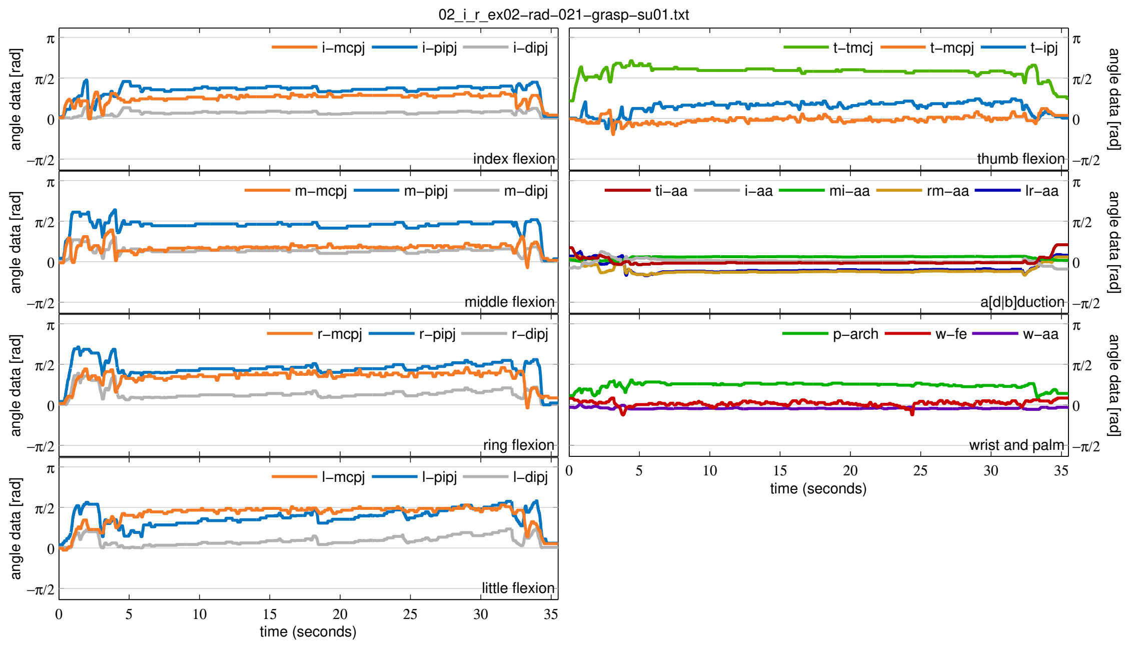


Figure 3.9: Exemplary angle trajectories recorded in *object use and interaction*. In the motion, the actor picked up a pen, opened it, wrote a short sentence, capped the pen again, and put it back onto the table. Angle trajectories are grouped by finger and type of movement (flexion/extension, adduction/abduction).

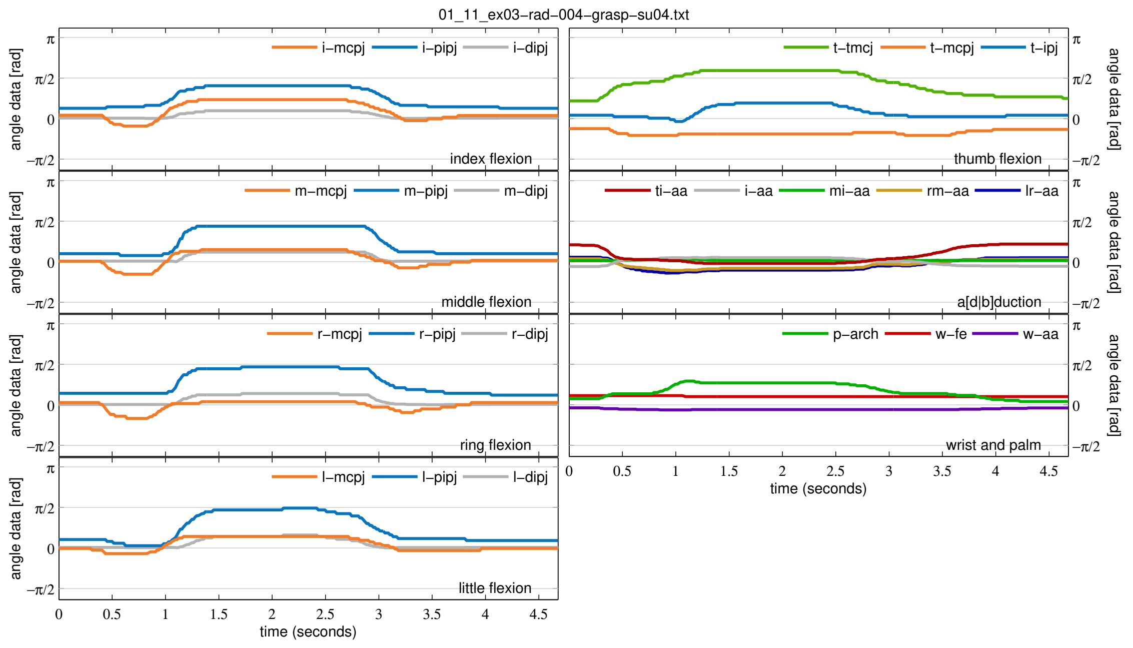


Figure 3.10: Exemplary angle trajectories recorded in *controlled object transport*. The actor picked up a tennis ball using a power sphere grasp (grasp ID 11), held it for a few moments and put the ball back down. Angle trajectories are grouped by finger and type of movement (flexion/extension, adduction/abduction).

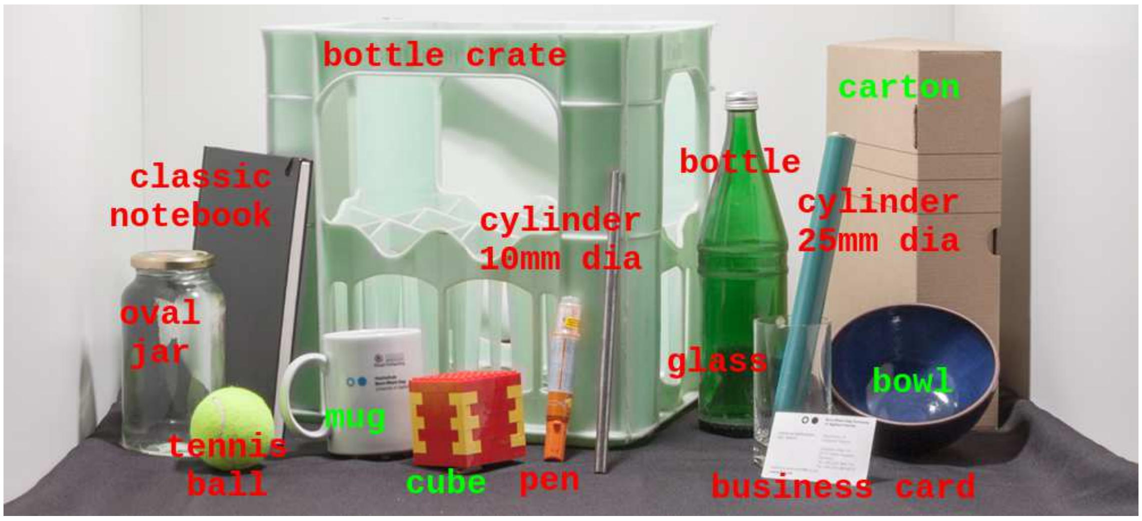


Figure 3.11: Objects used for grasping: classic notebook, oval jar, tennis ball, mug, cube, bottle crate, small cylinder, pen, bottle, glass, business card, bowl, cylinder large, carton (7.6 $\times$ 26 $\times$ 37.5 cm^3). Only objects in red were used in controlled transport.

On the choice of grasp types in controlled object transport

Several fields of research (e.g. biomechanics, robotics, medicine) have introduced grasp taxonomies for grouping different grasps by common criteria seeking simplification of the hand's complex prehensile capabilities. Due to the wide range of application and a resulting lack of consensus in naming and classifying grasp types Feix et al. [Fei+15] collected and consolidated the vast amount of grasp examples found in literature. Their taxonomy comprises 33 grasp types grouped into 17 basic types by merging equivalent grasps. The chosen taxonomy defines a grasp as *a static configuration of one hand able to securely hold an object* (in that hand). This explicitly rules out intrinsic movements in the hand during grasp, bi-manual interaction and gravity-dependent grasps such as an object lying in equilibrium on a flat hand.

Out of the list of 17 basic grasp types we have chosen to cover 13, omitting specialty configurations such as holding chopsticks (tripod variation), a pair of scissors (distal type), a cigarette between index finger and middle finger (abduction grip), and holding a lid of a bottle after unscrewing it (lateral tripod). Some of the basic grasp types were covered more often, hence extending the number of grasp types to 25 also using different sized objects. For a complete list of grasps and objects used for controlled transport see table 3.3 and figure 3.11. Additionally, figures A.1 through A.3 depict all objects used in the creation of the database individually.

3.7 Cutting of the recorded scenarios into motion trials

As reported above, the CyberGlove's switch state was used to differentiate between task executions and unwanted motions during the small breaks between tasks. This state was used to automatically separate the full scenario motion sequences into individual motion trials. After cutting, each motion trial contains exactly one grasp movement.

The final database contains approximately 2200 grasp motions of ten different actors interacting with 15 different objects represented by 25 grasp types (13 basic grasp types). Each motion is annotated with the object that was grasped, and – for controlled transport – the grasp type that was used, or – for object use and interaction – the action performed on the data base file name pattern). These annotations are encoded in the motion trials' file name. Section 4.1 describes how to decode the file names.

Power grasp, palm opposition

ID	Grasp name	Object
Virtual fingers: P, 2-5, –		
1	Large diameter	Bottle (8 cm diameter)
2	Small diameter	Cylinder 25 mm diameter
3	Medium wrap	Bottle crate
10	Power disk	Oval jar (lid, 7 cm diameter)
11	Power sphere	Tennis ball (64 mm diameter)

ID	Grasp name	Object
Virtual fingers: P, 3-5, 2		
17	Index finger extension	Cylinder 25 mm diameter
Virtual fingers: P, 2-5, –		
4	Abducted thumb	Cylinder 25 mm diameter
15	Fixed hook	Bottle crate
30	Palmar	Classic notebook (15 mm thick)

Power grasp, pad opposition

ID	Grasp name	Object
Virtual fingers: 1, 2, –		
31	Ring	Glass (62 mm diameter)
Virtual fingers: 1, 2-3, –		
28	Sphere 3 finger	Tennis ball (64 mm diameter)
Virtual fingers: 1, 2-4, –		
18	Extension type	Classic notebook (15 mm thick)
26	Sphere 4 finger	Tennis ball (64 mm diameter)

Intermediate grasp, side opposition

ID	Grasp name	Object
Virtual fingers: 1, 2, –		
16	Lateral	Business card
32	Ventral	Cylinder 25 mm diameter

Precision grasp, pad opposition

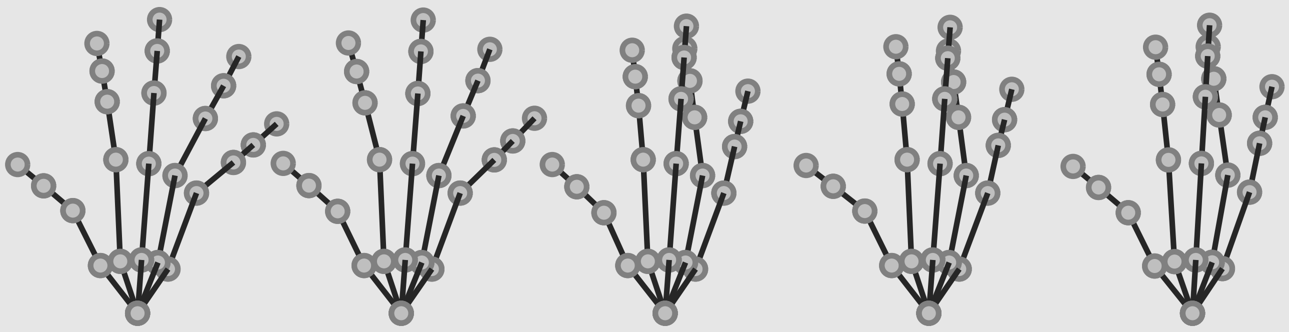
ID	Grasp name	Object
virtual fingers: 1, 2, –		
9	Palmar pinch	Business card
33	Inferior pincer	Tennis ball (64 mm diameter)
Virtual fingers: 1, 2-3, –		
8	Prismatic 2 finger	Cylinder 10 mm diameter
14	Tripod	Bottle (cap, 3 cm diameter)
Virtual fingers: 1, 2-4, –		
7	Prismatic 3 finger	Cylinder 10 mm diameter
27	Quadpod	Tennis ball (64 mm diameter)
Virtual fingers: 1, 2-5, –		
6	Prismatic 4 finger	Cylinder 10 mm diameter
13	Precision sphere	Tennis ball (64 mm diameter)

ID	Grasp name	Object
Virtual fingers: 1, 2-5, –		
22	Parallel extension	Classic notebook (15 mm thick)

Precision grasp, side opposition

ID	Grasp name	Object
Virtual fingers: 1, 3, –		
20	Writing tripod	Pen (2 cm diameter)

Table 3.3: List of grasps in controlled transport grouped by basic grasp and opposition type (vertically grouped / split horizontally) and thumb flexion (left hand side: thumb abduction, right hand side: thumb adduction). The list includes the grasp type's number and name based on Feix et al. [Fei+15] and the objects grasped.



4. File formats

4.1 File name pattern

This section provides an overview of the file naming convention for camera calibration files and database motion files along with illustrational examples.

4.1.1 Camera calibration files

The calibration file name is identical to the file name of the rectified image it refers to, except that its extension is `.xml`. Rectified images are stored with "`-rect`" appended to the original image file name: `<imageFilename>-rect.jpg`.

■ **Example 4.1** If the original image taken of a actor's hand is named `img123.jpg`, the rectified image will be `img123-rect.jpg` and the calibration file `img123-rect.xml`. ■

4.1.2 Database motion files

Each database motion file name contains information about the object grasped or manipulated (objectID), the recorded scenario (exNo), the type of data recorded (datatype, either `rad` or `raw`), a linear counter (trialID), the actor who performed the task (actorID), the action performed (actionID, only in the second scenario), and the grasp used to grasp the object (graspID, only in the third scenario). The mapping of an object's ID to its name can be found in table 4.1. The table also shows which grasp ID was used for each object (for controlled object transport). An overview of the grasp IDs and what concrete type of grasp they represent is given in table 3.3. Table 4.2 may be used to help interpret action IDs.

For uncontrolled object transports (first scenario, see 3.6.1) the file names are set-up as follows: `<objID>_ex<exNo>-<datatype>-<trialID>-grasp-su<actorID>.txt`.

■ **Example 4.2** The file name `01_ex01-raw-002-grasp-su01.txt` translates to: Actor 1 (su01) grasped a tennis ball (01) in the first scenario (ex01). The trial counter is 2 (002) and data was recorded in radians (rad). ■

For object use and interaction (second scenario, see 3.6.2) the file names are set up as follows: `<objID>_<actionID>_ex<exNo>-<datatype>-<trialID>-grasp-su<actorID>.txt`.

Object ID	Object name	Grasp ID(s)
1	Tennis ball (64 mm diameter)	11, 13, 26, 27, 28, 33
2	Pen (2 cm diameter)	20
3	Bottle (8 cm diameter)	1
4	Mug	–
5	Glass (62 mm diameter)	31
6	Bowl	–
7	Cube	–
8	Business card	9, 16
9	Oval jar	10
10	Bottle crate	3, 15
11	Classic notebook (15 mm thick)	18, 22, 30
12	Cylinder 10 mm diameter	6, 7, 8
13	Cylinder 25 mm diameter	2, 4, 17, 32
14	Carton lying flat	–
15	Carton standing ($7.6 \times 26 \times 37.5$ cm^3)	–
16	Bottle (cap)	14

Table 4.1: Mapping between object ID, object name, and grasp ID(s) according to [Fei+15] (when applicable).

Action ID	Action	Leading hand
o_r	open	right
s_r	close	right
f_r	close tightly	right
i_r	interact	right

Action ID	Action	Leading hand
o_l	open	left
s_l	close	left
f_l	close tightly	left

Table 4.2: Mapping of action abbreviations (action ID) to the action represented (action) and the hand performing the action (leading hand).

■ **Example 4.3** The file name `01_i_r_ex02-raw-025-grasp-su01.txt` translates to: Actor 1 (su01) right handedly interacted (i_r) with a ball (01), i.e. threw it, in the second recording scenario (ex02). The trial counter is 25 (025) and raw sensor data was recorded (raw). ■

For controlled object transports (third scenario, see 3.6.3) the file names are set-up as follows: `<objID>_<graspID>_ex<exNo>-<datatype>-<trialID>-grasp-su<actorID>.txt`.

■ **Example 4.4** The file name `01_11_ex03-1-rad-001-grasp-su04.txt` translates to: Actor 4 (su04) used the power sphere grasp (11) to hold a tennis ball (01) in the third recording scenario (ex03-1). The trial counter is 1 (001) and data was recorded in radians (rad). ■

4.1.3 Database skeleton files

Mapping of the recorded joint angles to a kinematic chain model is done via a rudimentary skeleton file. File names for skeleton files are arbitrary except for the file name extension. The file name extension for skeleton files is `.xmlskel`.

4.2 File content

The following part details the structure and content of the saved files ranging from the initial calibration process to the final cut motion clips.

4.2.1 Camera calibration files

The camera calibration file contains information about the calibration parameters of the camera at the time the image of the actor's hand was taken 3.4. Additionally, it contains the lengths (in pixels and mm) of the hand's bone segments (starting from the metacarpals outwards) as well as the 2D positions (in pixels) of the joints (in the rectified image). The calibration file is stored as an an .xml-file structured as shown in listing 4.1.

Listing 4.1: Structure of the camera calibration .xml-files. Data entries of arrays of any kind longer than two entries are here represented by a description (e.g. 'matrix-entries').

```
<?xml version="1.0"?>
<opencv_storage>
  <camera>
    <intrinsic>
      <calibration_matrix type_id="opencv-matrix">
        <rows>3</rows>
        <cols>3</cols>
        <dt>d</dt>
        <data>matrix entries</data>
      </calibration_matrix>
      <distortions type_id="opencv-matrix">
        <rows>5</rows>
        <cols>1</cols>
        <dt>d</dt>
        <data>matrix entries</data>
      </distortions>
    </intrinsic>
    <extrinsic>
      <translation_vector type_id="opencv-matrix">
        <rows>3</rows>
        <cols>1</cols>
        <dt>d</dt>
        <data>matrix entries</data>
      </translation_vector>
      <rotation_vector type_id="opencv-matrix">
        <rows>3</rows>
        <cols>1</cols>
        <dt>d</dt>
        <data>matrix entries</data>
      </rotation_vector>
    </extrinsic>
  </camera>
  <homography>
    <homography_matrix type_id="opencv-matrix">
      <rows>3</rows>
      <cols>3</cols>
      <dt>d</dt>
      <data>matrix-entries</data>
    </homography_matrix>
    <scale_factor>5</scale_factor>
    <board_size>1501 1001</board_size>
  </homography>
 <!-- resumed -->
  <hand>
    <right>
```

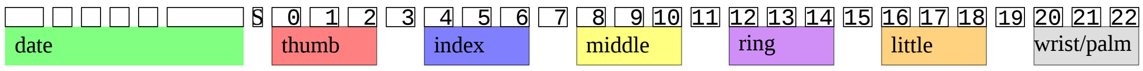


Figure 4.1: Visualisation of data ordering in a data row of a motion file. The numbers in the top rectangles represent sensor IDs (0, ..., 22). These are consistent with tables 4.3 and 3.1. In the bottom part, values of the row above are grouped according to meaning (date, thumb, fingers, and wrist/palm). Ungrouped entries do not explicitly belong to one of the groups (such as abduction/adduction sensors) and are left standing solitary.

```
      <image_based_measurement>
        <raw>
          <coordinates_2d>joint coordinates in [pixel]</coordinates_2d>
          <lengths>segment lengths in [pixel]</lengths>
        </raw>
        <scaled>
          <coordinates_2d>joint coordinates in [mm]</coordinates_2d>
          <lengths>segment lengths in [mm]</lengths>
        </scaled>
      </image_based_measurement>
    </right>
  </hand>
</opencv_storage>
```

4.2.2 Database motion files

A motion file consists of several rows of data. Each row represents one frame (also called a sample). Entries in a row of data are separated by a blank. Each data row consists of 30 entries ranging from a timestamp over the state of the switch button to 23 recorded sensor readings. The timestamp includes the date (YYYY MM DD) and time (HH mm ss.sssss) for each sample. Seconds were recorded accurate to five decimal places. The switch state was saved as a boolean value represented by 0 (false / OFF) or 1 (true / ON). The sensor readings were recorded as 8-bit integer values for the raw sensor output and as floating point values when joint angles (in radians). Both versions were saved in separate files, one of which contains `raw` in its file name and the other one contains `rad` (cf. section 4.1.2). The concrete data ordering of each data row is visualised in figure 4.1 and detailed in table 4.3.

■ **Example 4.5** Exemplary data row (raw 8-bit data). An extract (up to the first nine sensor entries) of an exemplary data row of a raw-valued motion recording looks as follows:

```
   2014 09 12 08 49 21.71200 1 101 114 128 107 136  94   0   0 119 ...
   YYYY MM DD HH mm ss.sssss S   0   1   2   3   4   5   6   7   8 ...
```

■

4.2.3 Database skeleton files

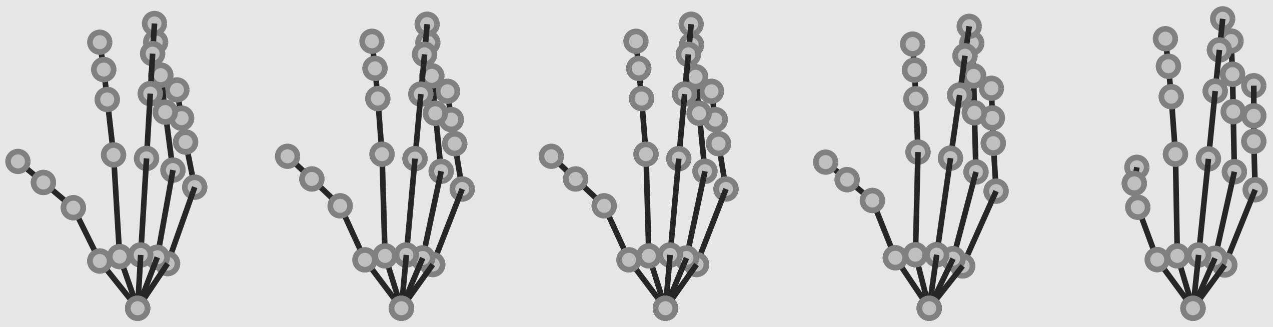
Skeleton files contain information on the underlying kinematic chain model of the hand. It is saved in xml-format. The hand model's topology is saved in a hierarchical manner: The underlying structure is a tree rooted in the wrist. Joints represent nodes in the tree. Each node contains the joint's name, its id within the structure, its position, the limits of possible motions along three (canonical) axes and its child node(s). Apart from the kinematic chain, the file also includes information on the units used for representing angles and lengths. An exemplary (shortened) database skeleton file is printed in listing 4.2.

Data	Format	Sensor no.		Key
year	(YYYY)			1
month	(MM)			2
day	(DD)			3
hour	(HH)			4
minute	(mm)			5
seconds	(ss.sssss)			6
Switch	(bool)			7
T-TMJ	(float)	0		8
T-MCPJ	(float)	1		9
T-IPJ	(float)	2		10
TI-AA	(float)	3		11
I-MCPJ	(float)	4		12
I-PIPJ	(float)	5		13
I-DIPJ	(float)	6	*	14
I-AA	(float)	7	**	15
M-MCPJ	(float)	8		16
M-PIPJ	(float)	9		17
M-DIPJ	(float)	10	*	18
MI-AA	(float)	11		19
R-MCPJ	(float)	12		20
R-PIPJ	(float)	13		21
R-DIPJ	(float)	14	*	22
RM-AA	(float)	15		23
L-MCPJ	(float)	16		24
L-PIPJ	(float)	17		25
L-DIPJ	(float)	18	*	26
LR-AA	(float)	19		27
P-Arch	(float)	20		28
W-FE	(float)	21		29
W-AA	(float)	22		30

Table 4.3: List of a motion files' data row entries. It contains the data elements saved (data) along with the format used for representing its value, the sensor number (no.) where applicable, and a key specifying the data's position in the data row. Sensor denotations in the column 'data' and sensor numbers are consistent with table 3.1 and figure 3.1b. * marks sensors not present in an 18-sensor data glove. Their value will be zero (when recording 8-bit raw data) or inferred (when recording angle data). ** marks unsupported sensors the value of which will always be 0.

Listing 4.2: Structure of the database skeleton .xmlskel-files. The structure presented below is only an excerpt of an original .xmlskel-file. It contains full information on the wrist (root) and the thumb trapeziometacarpal joint (th-tmc). The remaining joints of the thumb are here shortened to name and id. The other fingers' details are completely omitted and only represented textually by naming their joints' name and id. Their real structure is similar to the structure of the thumb.

```
<skeleton>
  <units>
  <angle>deg</angle>
  <length>mm</length>
  </units>
  <!-- start root (wrist) -->
  <joint>
    <name>root</name>
    <id>1</id>
    <position>0 0 0</position>
    <limits>
      <rotx>-180 180</rotx>
      <roty>-180 180</roty>
      <rotz>-180 180</rotz>
    </limits>
    <!-- start thumb -->
    <joint>
      <name>th-tmc</name>
      <id>2</id>
      <position>27 -31 0 </position>
      <limits>
        <rotx>-25 35</rotx><!--flexion/extension-->
        <roty>0 0</roty>
        <rotz>-30 60</rotz><!--adduction/abduction-->
      </limits>
      <joint>
        <name>th-mcp</name>
        [id (3), position, limits]
        <joint>
          <name>th-ip</name>
          [id (4), position, limits]
          <joint>
            <name>th-tip</name>
            [id (5), position, limits]
          </joint>
        </joint>
      </joint>
    </joint> <!-- end thumb -->
    <!-- start index finger -->
    <joint>
      [if-cmc (6), if-mcp (7), if-pip (8), if-dip (9), if-tip (10)]
    </joint> <!-- end index finger -->
    <!-- start middle finger -->
    <joint>
      [mf-cmc (11), mf-mcp (12), mf-pip (13), mf-dip (14), mf-tip (15)]
    </joint> <!-- end middle finger -->
    <!-- start ring finger -->
    <joint>
      [rf-cmc (16), rf-mcp (17), rf-pip (18), rf-dip (19), rf-tip (20)]
    </joint> <!-- end ring finger -->
    <!-- start little finger -->
    <joint>
      [lf-cmc (21), lf-mcp (22), lf-pip (23), lf-dip (24), lf-tip (25)]
    </joint> <!-- end little finger -->
  </joint> <!-- end root -->
</skeleton>
```



5. Tools

5.1 C++ tools

Mainly for the purpose of reproducibility, this section presents two tools written in C++ for measuring distances in the calibrated / rectified images of the actors' hands and for verifying previously saved measurements. Both tools were used during the creation of the database, e.g. for creating custom hand skeleton models in form of a kinematic chain .xmlskel-file as described in section 4.2.3.

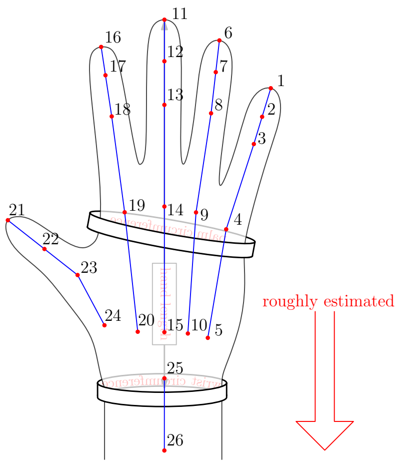
Both tools use OpenCV [CGI17] (tested release OpenCV 3.1.0) for loading and displaying images and zenity [Zen17] for creating simple graphical user interfaces. The measurement verification tool additionally uses TinyXML-2 [Tho17] for parsing the files created in the hand measuring process and Eigen3 [JG17] for matrix operations.

5.1.1 Measuring tool

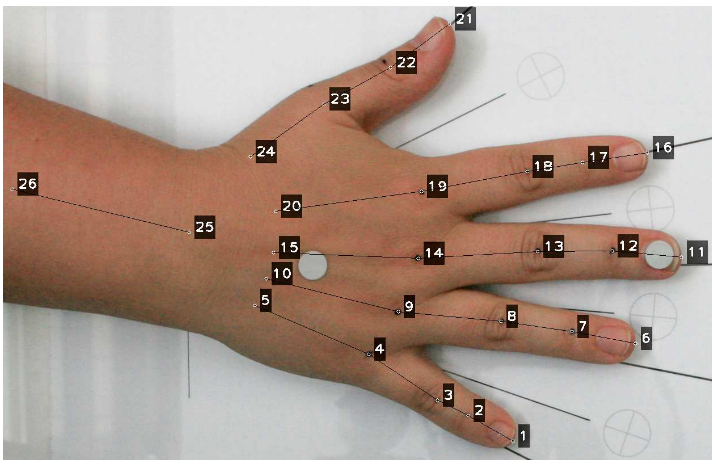
The measuring tool can be used to generate individual hand skeletons by providing means to measure each bone's length in a rectified image. It uses OpenCV for appending the measured hand data (2D coordinates of the joints and lengths of their connecting segments in pixels and in millimetres). This data is saved when pressing '`s`' by appending it to the loaded image's camera calibration file (see listing 4.1). It will not replace previously saved hand measurement data, which – for this reason – would have to be removed manually. The measuring tool allows for measuring the hand segments (e.g. bones). Clicking onto the marked landmarks representing the joint structure will add a point, pressing `<backspace>` will remove the last added point. The points have to be clicked in a predefined order which is depicted in figure 5.1(a). The image representation resulting from clicking all necessary joints is shown in figure 5.1(b). `<Esc>` quits the program.

5.1.2 Measurement verification tool

The main purpose of the measurement verification tool is to visually verify and inspect results from the measuring tool (see figure 5.2). Additionally it provides the possibility to compute the distance between pairs of point in the image. Pressing '`m`' enters that measuring mode. A different image can be loaded pressing '`l`'. `<Esc>` quits the program.



(a) Ordering and approximate locations of landmarks.



(b) Image resulting from clicking all necessary landmarks.

Figure 5.1: Measuring hand segments using the measuring tool. (a) shows the ordering (and approximate locations of felt-marked landmarks) to be clicked in the measuring tool. ‘roughly estimated’ indicates that these points are not marked on the skin. Point 25 is located centred over the wrist and the segment between points 25 and 26 should indicate the direction of the arm. (b) shows the image resulting from clicking all necessary landmarks. Segments are drawn as lines and landmarks are labelled with their respective number.

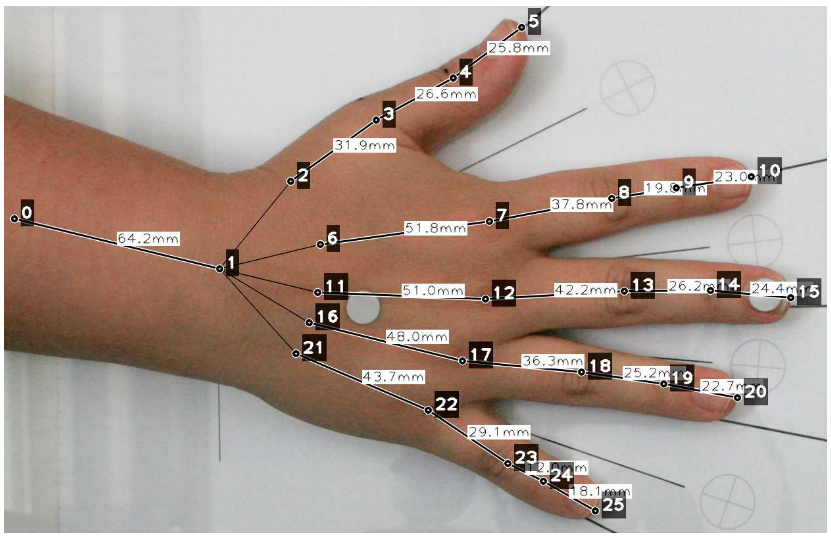


Figure 5.2: Result of loading an image together with the saved hand data in the measurement verification tool. Landmarks are numbered in reverse order with respect to the measuring tool. Segments are again displayed as lines along with their length in mm.

5.2 MATLAB tools

In the following, this section describes how to load and visualise the recorded motion data in MATLAB using the MATLAB tools provided together with the recorded database files. All tools were programmed and tested using MATLAB 2012b.

5.2.1 Parsing tool

The parsing tool provides means to load a hand skeleton and a database motion file. In case of loading an angle based database motion file the parser also calculates the hand joints' trajectories via forward kinematics.

The commands for reading an exemplary skeleton file and motion database file are:

```
skel = readSkeleton('default.xmlskel');
mot  = loadDBKSmotion('01_11_ex03-rad-004-grasp-su04.txt', skel, 'full');
```

Skeleton file parser `readSkeleton`

The skeleton file parser reads in an `.xmlskel` file as described in section 4.2.3 and converts it to a more suitable MATLAB structure. The MATLAB-struct returned by the skeleton file parser has the following fields:

```
skel =

  numberofjoints: 25                    % number of joints
          joints: [25x1 struct]         % joints representing nodes in the skeleton
                                        %   (tree rooted in the wrist) structure
      jointnames: {25x1 cell}           % joint names
         namemap: {25x3 cell}           % cell array mapping struct IDs to joint IDs
                                        %   to joint names (struct IDs are obsolete)
           paths: {5x1 cell}            % contains paths for drawing the skeleton
                                        %   (sequence of joint IDs)
       angleunit: 'deg'                 % angle unit (rad or deg)
      lengthunit: 'mm'                  % length unit (mm or ...)
           scale: 1                     % scale factor for the skeleton
        filename: 'default.xmlskel'     % filename of this skeleton
        filetype: 'XMLSKEL'             % file type of this skeleton
```

The field `skel.joints` contains the hierarchy information of the skeleton's underlying kinematic chain. Each individual `joint` represents a node in the chain, pairs of connected joints represent bones. Only the root joint has more than one child (it has five); all other joints have only one child. End effectors (finger tips) have no children.

```
skel.joint(8) =

              jointname: 'if-pip'       % name of the joint
                     id: 8              % ID of the joint
               children: 9              % [1xn] list of child joint IDs
                 parent: 7              % parent joint ID
                 offset: [3x1 double]   % offset in position of this joint to its
                                        %   parent
               position: [3x1 double]   % absolute position of the this joint
  localcoordinatesystem: [3x3 double]   % columns of this matrix represent the
                                        %   local coordinate system
                    dof: [1x3 logical]  % information if axis i is a dof, i.e.
                                        %   dof(i)=1 => axis i is a dof,
                                        %   dof(i)=0 => axis i is not
                 limits: [3x2 double]   % limits in [deg] for each dof
```

It should be noted that the skeleton file only represents a simple underlying kinematic structure for the joint angles recorded by the CyberGlove. Figure 5.3 displays both, the plain flat outstretched hand represented by our skeleton file, and an intermediate grasping pose. Table 5.1 gives details on the mapping of CyberGlove sensors to skeleton hand joints.

Database motion file reader `loaDBKSmotion`

The database motion file reader combines the recorded angular joint data and the skeletal kinematic chain model of the hand into a succession of hand joint configurations suitable for animation. There are three options for loading a a motion file:

- `'plain'` loads the motion without calculating 3D joint trajectories.
- `'full'` loads the motion and computes all 3D joint trajectories.
- `'fixed-root'` loads the motion simulating the wrist did not move when calculating 3D joint trajectories.

The database motion file reader returns a MATLAB-struct of the following form:

```
mot =

          numberofangles: 23                 % number of angles in the motion
          numberofframes: 282                % number of frames (#frames)
            samplingrate: 60.2564            % sampling rate (inverse of frame time)
              timestamps: [282x6 double]     % time stamps
             elapsedtime: [282x1 double]     % relative time stamps (time elapsed
                                             %   since the first frame)
            switchstates: [282x1 double]     % state of the switch (ON|OFF)
               angledata: [282x23 double]    % succession of joint angles
             sensornames: {23x1 cell}        % name identifier for each sensor
                  active: [18x1 double]      % sensor IDs marking recorded angles
                                             %   (opposed to computed angles)
                inactive: [5x1 double]       % sensor IDs marking computed angles
                                             %   (opposed to recorded angles)
               angleunit: 'rad'              % angle unit (rad or deg)
                filename: [1x35 char]        % name of the motion source file
       jointtrajectories: {25x1 cell}        % position of the joints over time
  localcoordinatesystems: {25x282 cell}      % local coordinate system of the joints
```

Here, each entry of the field `mot.jointtrajectories` contains the calculated 3D trajectory of a specific joint, e.g. `mot.jointtrajectories{8}` contains the 3D trajectory of the index finger's proximal interphalangeal joint ('if-pip'). The trajectories are calculated based on the recorded (and calculated) joint angles using forward kinematics. Hence, each joint trajectory has the size $3 \times$ #frames. The rows of the field `mot.localcoordinatesystems` contain the succession of each joint's local coordinate system over time; local coordinate systems vary with the angular progression in the joints. Each coordinate system is saved as a 3×3 matrix, the columns of which represent the three axes of the coordinate system. Its origin is the current position of the associated joint (see figure 5.3).

The difference in number between joint trajectories and sensor angles results from different reasons: On the one hand, some recorded sensor angles combine into one joint of several degrees of freedom. This is the case for the six adduction/abduction angles and the angle describing arching the palm (compare table 5.1). On the other hand, the model hand skeleton we introduced has additional structural joints that do not reflect any angle data recordings. This is true for four carpometacarpal joints and five finger tips (see figures 3.1 and 5.3); the latter acting as end effectors. In sum, this is how 23 recorded (and computed) joint angles are represented by 25 joints in the kinematic chain.

5.2.2 Visualisation tool

This section describes how to transform the recorded joint angle data into animation data using `calcJointTrajectories`. It also shows how this motion data can be animated using the `animate_single` or `animateGUI` commands provided with this tool.

Calculating 3D joint trajectories `calcJointTrajectories`

In order to animate the hand from angle data, the 3D trajectories for each joint need to be computed. This is done via forward kinematics using the recorded angle data for each joint. Apart from loading a motion by calling the database motion file reader as explained in section 5.2.1, it is

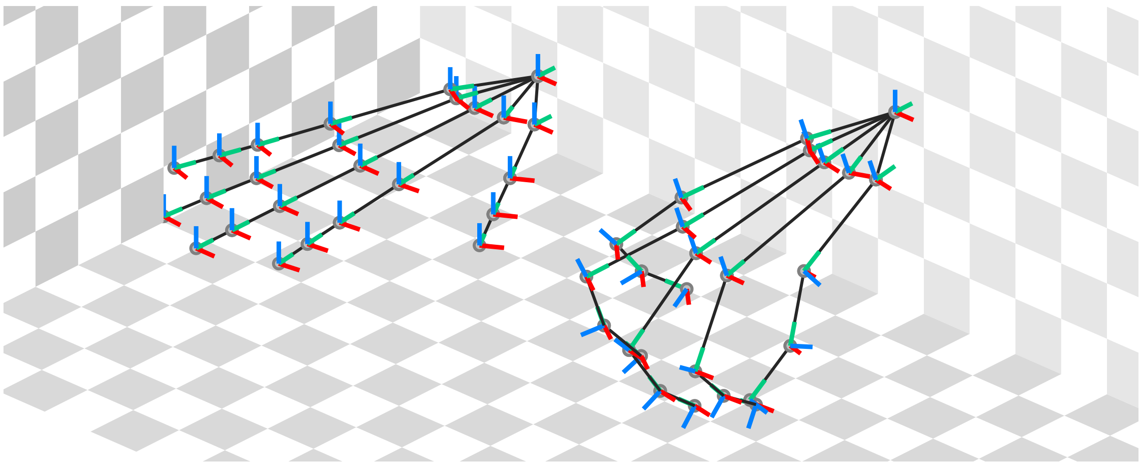


Figure 5.3: Flat hand pose of our default hand skeleton (left) and intermediate grasp pose (frame 150 of the above loaded motion, right) along with each joint's local coordinate system. The axes point away from the joints and are colour-coded red (x-axis), green (y-axis), and blue (z-axis).

Sensor ID	**Joint ID**	**Axis**	**Sensor name**	**Joint name**
1	2	z	T-TMCJ	th-tmc
2	3	x	T-MCPJ	th-mcp
3	4	x	T-IPJ	th-ip
4	3	z	TI-AA	th-mcp
5	7	x	I-MCPJ	if-mcp
6	8	x	I-PIPJ	if-pip
7	9	x	I-DIPJ	if-dip
8	7	—	I-AA	if-mcp
9	12	x	M-MCPJ	mf-mcp
10	13	x	M-PIPJ	mf-pip
11	14	x	M-DIPJ	mf-dip
12	7	z	MI-AA	if-mcp
13	17	x	R-MCPJ	rf-mcp
14	18	x	R-PIPJ	rf-pip
15	19	x	R-DIPJ	rf-dip
16	17	z	RM-AA	rf-mcp
17	22	x	L-MCPJ	lf-mcp
18	23	x	L-PIPJ	lf-pip
19	24	x	L-DIPJ	lf-dip
20	22	z	LR-AA	lf-mcp
21	21	—	P-Arch	lf-cmc
22	1	x	W-FE	root
23	1	z	W-AA	root

Table 5.1: Mapping of CyberGlove sensors to joints of our default hand skeleton. The column 'axis' specifies the local axis of rotation of each joint when applying the sensors' recorded angle data. Rotation around the local x-axis represents flexion (positive angle) and extension (negative angle). Similarly, a rotation around the local z-axis represents abduction and adduction of the corresponding joint. Sensor recordings whose axis is marked with '—' are currently discarded.

possible to calculate the joint trajectories in a more flexible way using the functionality provided by the `calcJointTrajectories` function. This function allows to completely deactivate a joint for animation by specifying (by ID) which of the recorded sensors not to take into account. An exemplary call with explanations is given in example 5.1. A helper functionality (`deactivatedJoints`) simplifies deactivation of an entire joint by returning the sensor IDs associated with a specific joint.

■ **Example 5.1** The following code snippet shows how to deactivate a specific joint for animation and then calculate the 3D joint trajectories of the recorded motion.

```
% get joint ID(s) of the middle finger's metacarpal joint (mf-mcp)
deactivatedJointIDs = name2index(skel.namemap, 'mf-mcp');
% get sensor ID(s) of the deactivated joints (mf-mcp)
deactivatedSensorIDs = deactivatedJoints( skel, mot, deactivatedJointIDs );
% compute 3D joint trajectories of the hand omitting the mf-mcp
mot = calcJointTrajectories ( mot, skel, deactivatedSensorIDs );
```

The variable `disabledSensors` will contain the sensor ID 9 which represents the sensor named ‘M-MCPJ’. For calculating the 3D joint trajectories, the function `calcJointTrajectories` will treat this sensor as deactivated by setting its angle data to zero. It is possible to deactivate more sensors/joint by simply concatenating them into a $[1 \times n$ double]-array, where n specifies the number of deactivated sensors. ■

Animation using `animate_single` and/or `animateGUI`

The visualisation tool provides two functions to display/animate the calculated motion data, `animate_single` and `animateGUI`. These are invoked as follows:

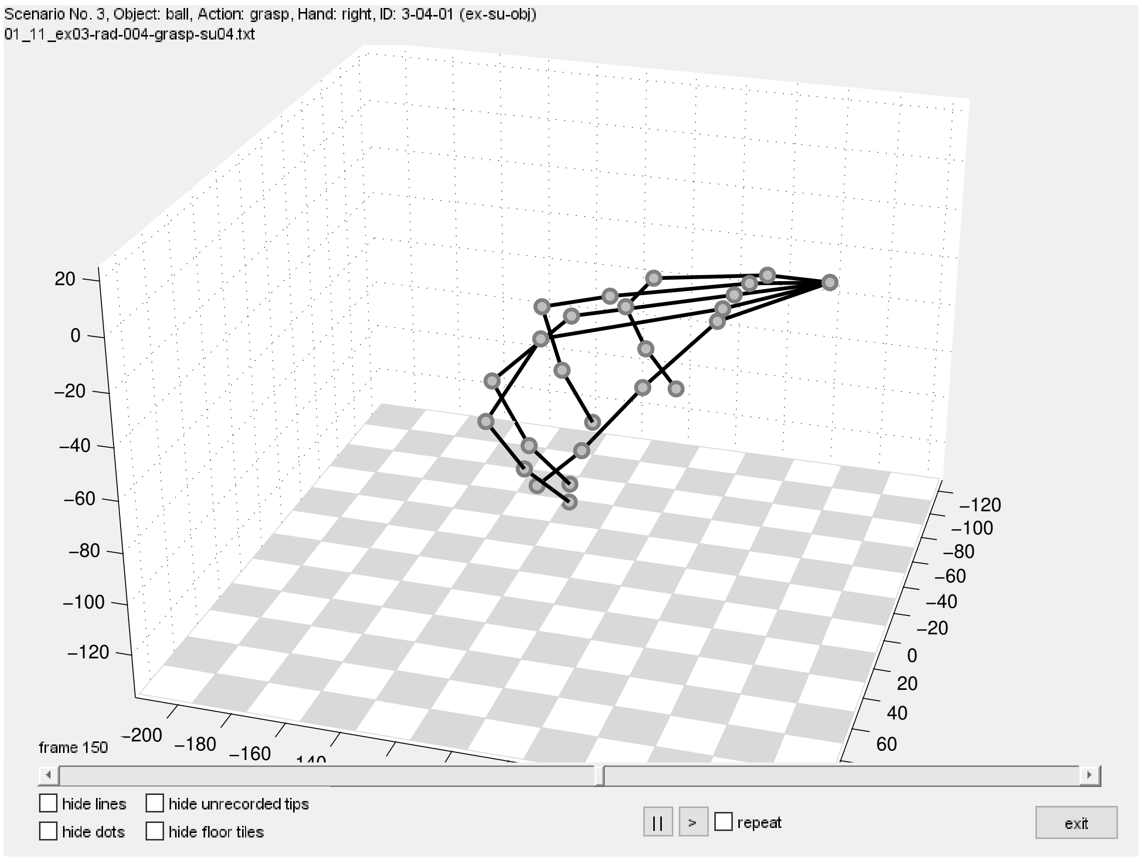
```
animate_single( skel, mot[, num_loops] );
```

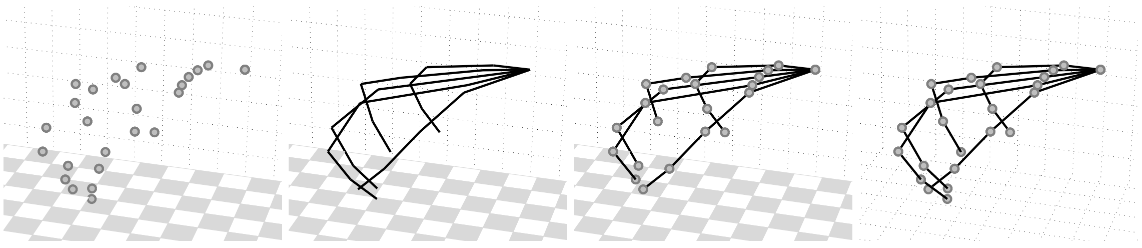
and

```
animateGUI( skel, mot );
```

The functions’ arguments `skel` and `mot` represent the previously explained skeleton and motion data MATLAB-structs (see section 5.2.1). `animate_single` can be optionally supplied with an additional argument specifying how many repetitions of the motion are to be played (`num_loops`).

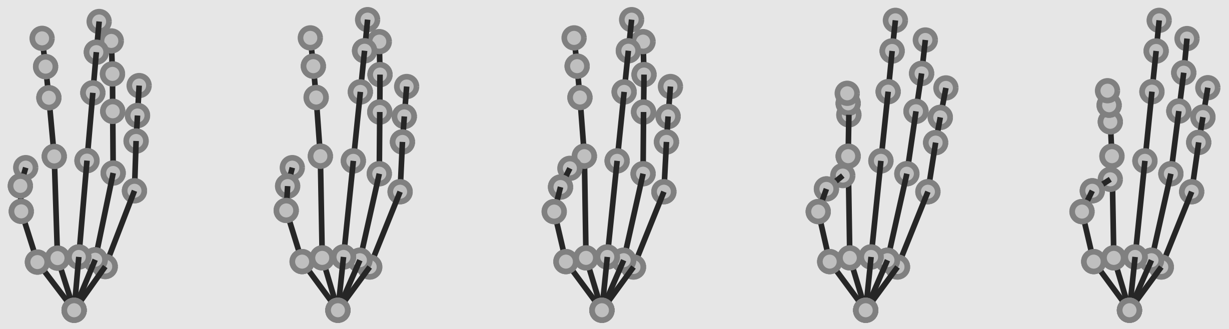
`animate_single` only offers a quick view of the motion data without much possibility of interaction – it can be rotated and displays the current frame in the animation sequence. `animateGUI` provides a more flexible view on the motion data. A screenshot of the GUI is depicted in figure 5.4. It is possible to navigate in the motion data’s time line (slider on the bottom) allowing for detailed inspection of the motion. The motion can also be played-back in a loop and paused (repeat checkbox, play, and pause button). The top right corner in the GUI displays information on the loaded motion data; among others, these are the scenario, the object used, and the type of interaction.


Scenario No. 3, Object: ball, Action: grasp, Hand: right, ID: 3-04-01 (ex-su-obj)
01_11_ex03-rad-004-grasp-su04.txt



(a) hide lines (b) hide dots (c) hide unrecorded (finger) tips (d) hide floor tiles

Figure 5.4: Screenshot of the overall animation GUI (top) and of the effect of activating the options ‘hide lines’, in order to hide the bone structure (a), ‘hide dots’, for hiding the joints (b), ‘hide unrecorded tips’ (c), and ‘hide floor tiles’ (d) on the visualisation of the hand and the ground (bottom row).



Bibliography

[Atz+12] Manfredo Atzori et al. ‘Building the Ninapro database: A resource for the biorobotics community’. In: *Proceedings of IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*. June 2012, pages 1258–1265. DOI: `10.1109/BioRob.2012.6290287` (cited on page 10).

[Bea+08] Philippe Beaudoin et al. ‘Motion-motif graphs’. In: *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’08. Dublin, Ireland: Eurographics Association, 2008, pages 117–126. ISBN: 978-3-905674-10-1 (cited on page 10).

[CMU13] CMU. *Carnegie Mellon University Graphics Lab: Motion Capture Database*. 2013. URL: `http://mocap.cs.cmu.edu` (cited on page 9).

[CGI17] Intel Corporation, Willow Garage and Itseez. *Open Source Computer Vision Library*. Accessed: 2017-11-22. 2017. URL: `https://opencv.org/` (cited on page 29).

[Cyb17] CyberGlove Systems. *Motion capture system web-site*. Accessed: 2017-11-21. 2017. URL: `http://www.cyberglovesystems.com/cyberglove-ii` (cited on page 9).

[EK05] Staffan Ekvall and Danica Kragic. ‘Grasp recognition for programming by demonstration’. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*. Apr. 2005, pages 748–753. DOI: `10.1109/ROBOT.2005.1570207` (cited on page 10).

[Fei+13] Thomas Feix et al. ‘A metric for comparing the anthropomorphic motion capability of artificial hands’. In: *IEEE Transactions on Robotics* 29.1 (Feb. 2013), pages 82–93 (cited on page 9).

[Fei+15] Thomas Feix et al. ‘The GRASP taxonomy of human grasp types’. In: *IEEE Transactions on Human-Machine Systems* 46.1 (Feb. 2015), pages 66–77. ISSN: 2168-2291. DOI: `10.1109/THMS.2015.2470657` (cited on pages 20, 21, 24).

[Gol+09] Corey Goldfeder et al. 'The Columbia grasp database'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2009, pages 1710–1716. DOI: 10.1109/ROBOT.2009.5152709 (cited on page 9).

[JG17] Benoît Jacob and Gaël Guennebaud. *Eigen – A template library for linear algebra*. Accessed: 2017-11-30. 2017. URL: http://eigen.tuxfamily.org/ (cited on page 29).

[JHS12] Sophie Jörg, Jessica K. Hodgins and Alla Safonova. 'Data-driven finger motion synthesis for gesturing characters'. In: *ACM Transactions on Graphics* 31.6 (Nov. 2012), 189:1–189:7. ISSN: 0730-0301. DOI: 10.1145/2366145.2366208 (cited on page 10).

[KI94] Sing Bing Kang and Katsushi Ikeuchi. 'Determination of motion breakpoints in a task sequence from human hand motion'. In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA)*. Volume 1. May 1994, pages 551–556. DOI: 10.1109/ROBOT.1994.351241 (cited on page 10).

[Lea17] Leap Motion. *Motion capture system web-site*. Accessed: 2017-11-21. 2017. URL: http://www.leapmotion.com (cited on page 9).

[LTK09] Sergey Levine, Christian Theobalt and Vladlen Koltun. 'Real-time prosody-driven synthesis of body language'. In: *ACM Transactions on Graphics* 28.5 (Dec. 2009), 172:1–172:10. ISSN: 0730-0301. DOI: 10.1145/1618452.1618518 (cited on page 10).

[Met17] Meta Motion. *Motion capture system web-site*. Accessed: 2017-11-21. 2017. URL: http://metamotion.com/gypsy/gypsy-motion-capture-system.htm (cited on page 9).

[MC12] Jianyuan Min and Jinxiang Chai. 'Motion graphs++: A compact generative model for semantic motion snalysis and synthesis'. In: *ACM Transactions on Graphics* 31.6 (Nov. 2012), 153:1–153:12. ISSN: 0730-0301 (cited on page 10).

[MAN15] Christos Mousas, Christos-Nikolaos Anagnostopoulos and Paul Newbury. 'Finger motion estimation and synthesis for gesturing characters'. In: *Proceedings of the 31st Spring Conference on Computer Graphics*. SCCG '15. Smolenice, Slovakia: ACM, 2015, pages 97–104. DOI: 10.1145/2788539.2788552 (cited on page 10).

[Mül+07] Meinard Müller et al. *Documentation Mocap Database HDM05*. Technical report CG-2007-2. Universität Bonn, June 2007 (cited on page 9).

[Opt17] OptiTrack. *Motion capture system web-site*. Accessed: 2017-11-21. 2017. URL: http://www.optitrack.com/ (cited on page 9).

[RLS13] Daniel Roetenberg, Henk Luinge and Per Slycke. *White paper: Xsens MVN: Full 6dof human motion tracking using miniature inertial sensors*. Technical report. Apr. 2013. URL: https://www.xsens.com/images/stories/PDF/MVN_white_paper.pdf (cited on page 9).

[ST17] Sensics and Russell M. Taylor II. *Virtual Reality Peripheral Network - Official Repo*. Accessed: 2017-11-30. 2017. URL: https://github.com/vrpn/vrpn/wiki/ (cited on page 12).

[Sho+13] Jamie Shotton et al. 'Real-time human pose recognition in parts from single depth images'. In: *Communications of the ACM* 56.1 (Jan. 2013), pages 116–124. ISSN: 0001-0782. DOI: 10.1145/2398356.2398381 (cited on page 9).

[Sto+16] Katharina Stollenwerk et al. 'Automatic temporal segmentation of articulated hand motion'. In: *Computational Science and Its Applications (ICCSA), Part II*. Springer International Publishing, 2016, pages 433–449. DOI: 10.1007/978-3-319-42108-7_33 (cited on page 10).

[Tay+01] Russell M. Taylor II et al. ‘VRPN: A device-independent, network-transparent VR peripheral system’. In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST ’01. Baniff, Alberta, Canada: ACM, 2001, pages 55–61. ISBN: 1-58113-427-4. DOI: `10.1145/505008.505019` (cited on page 12).

[Tho17] Lee Thomason. *A simple, small, efficient, C++ XML parser that can be easily integrating into other programs*. Accessed: 2017-11-22. 2017. URL: `http://www.grinninglizard.com/tinyxml2/` (cited on page 29).

[Vic17] Vicon. *Motion capture system web-site*. Accessed: 2017-11-21. 2017. URL: `http://www.vicon.com` (cited on page 9).

[Vir98] Virtual Technologies. *CyberGlove reference manual*. CG081998-2-1. Virtual Technologies, Inc. Palo Alto, Aug. 1998 (cited on page 12).

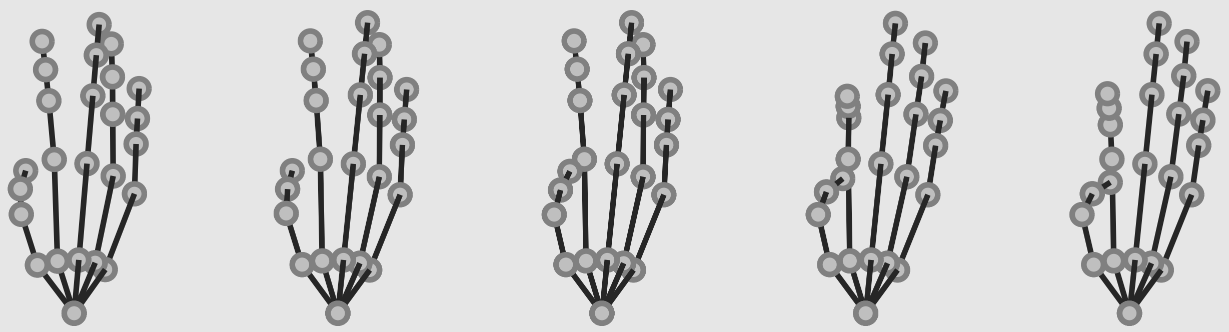
[VKK14] Anna Vögele, Björn Krüger and Reinhard Klein. ‘Efficient unsupervised temporal segmentation of human motion’. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’14. Copenhagen, Denmark: Eurographics Association, July 2014, pages 167–176 (cited on page 10).

[Whe+15] Nkenge Wheatland et al. ‘State of the art in hand and finger modeling and animation’. In: *Computer Graphics Forum* 34.2 (May 2015), pages 735–760. ISSN: 0167-7055. DOI: `10.1111/cgf.12595` (cited on page 9).

[Zen17] Zenity. *A tool that allows you to display GTK dialog boxes in commandline and shell scripts*. Accessed: 2017-11-21. 2017. URL: `https://wiki.gnome.org/Projects/Zenity` (cited on page 29).

[ZTH08] Feng Zhou, F. De la Torre and Jessica K. Hodgins. ‘Aligned cluster analysis for temporal segmentation of human motion’. In: *IEEE Conference on Automatic Face and Gestures Recognition*. Sept. 2008 (cited on page 10).

[ZTH13] Feng Zhou, F. De la Torre and Jessica K. Hodgins. ‘Hierarchical aligned cluster analysis for temporal clustering of human motion’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (Mar. 2013), pages 582–596. ISSN: 0162-8828 (cited on page 10).



Index

Appendix

A. Objects

The following pages list the objects used in the database by individual images. Most images provide an additional top or bottom view as well as a side view of the object. The images are sorted alphabetically by object name. The last image shows all objects together to illustrate the relative size of the objects.

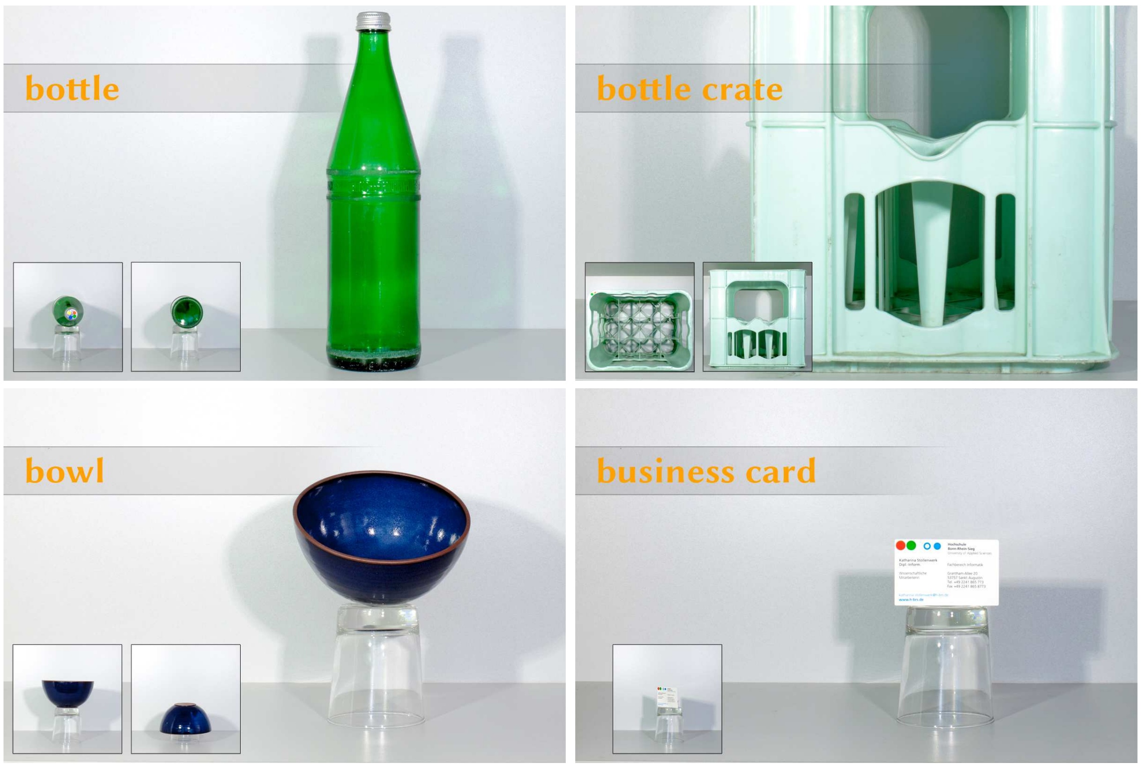


Figure A.1: Objects used for the creation of the database – a bottle, a bottle crate, a bowl, and a business card.

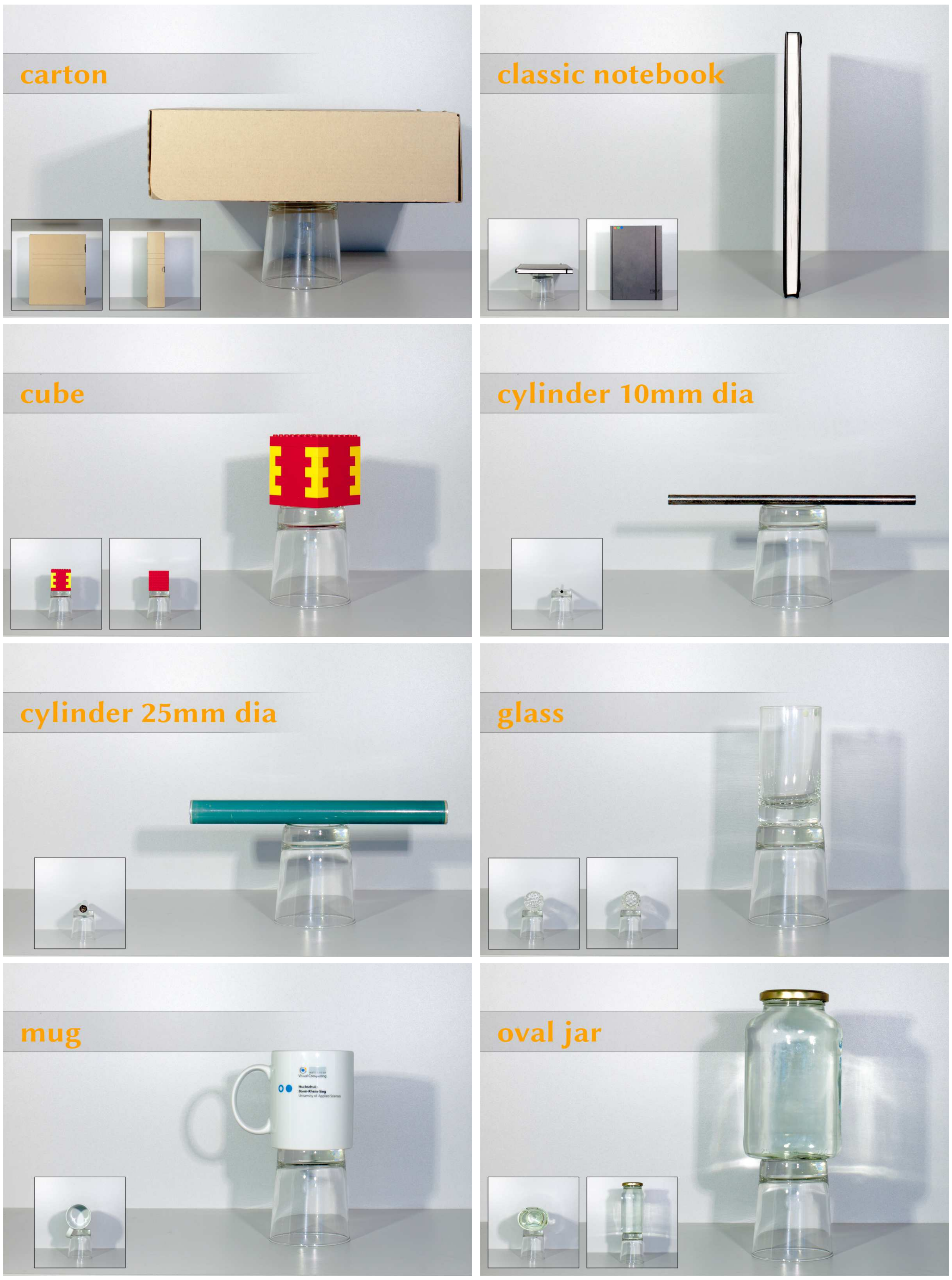


Figure A.2: Objects used for the creation of the database – a carton, which was used standing upright and lying flat, a classic notebook, a cube, a cylinder of smaller diameter, a cylinder of bigger diameter, a glass, a mug, and an oval jar.

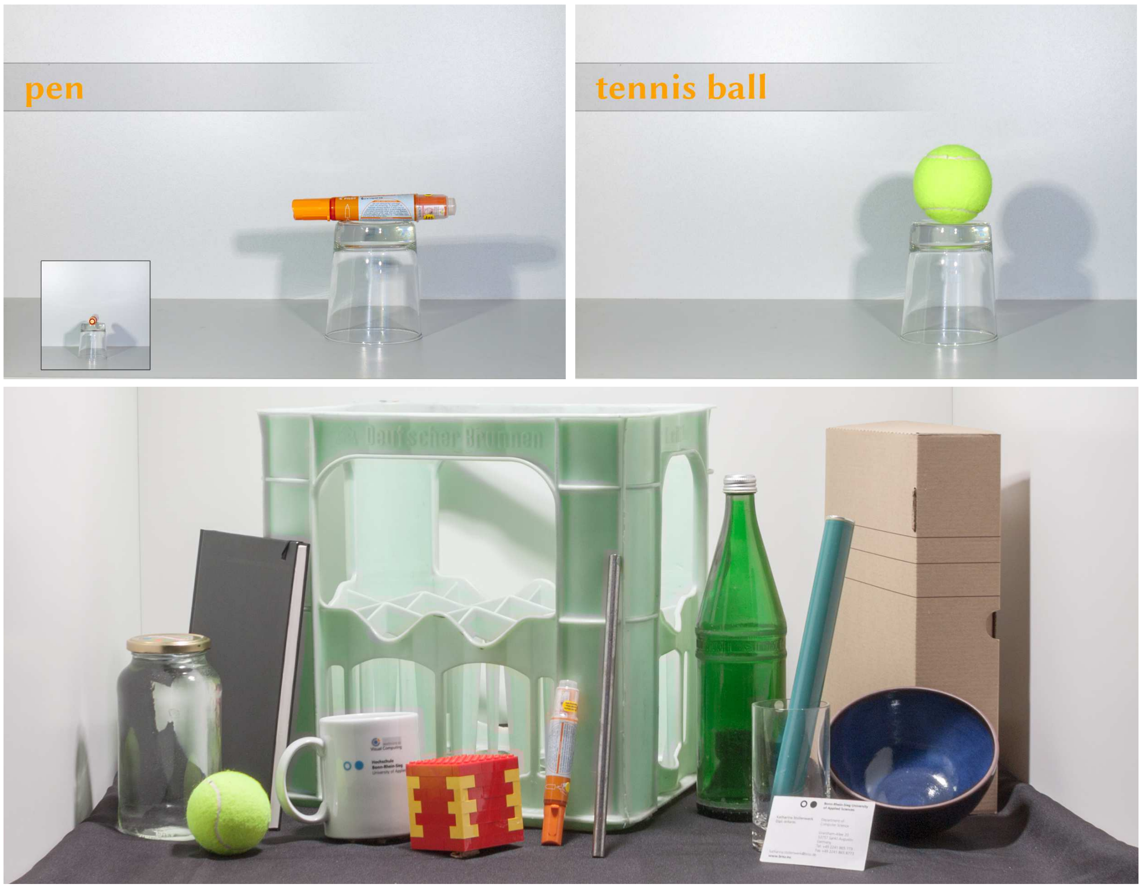
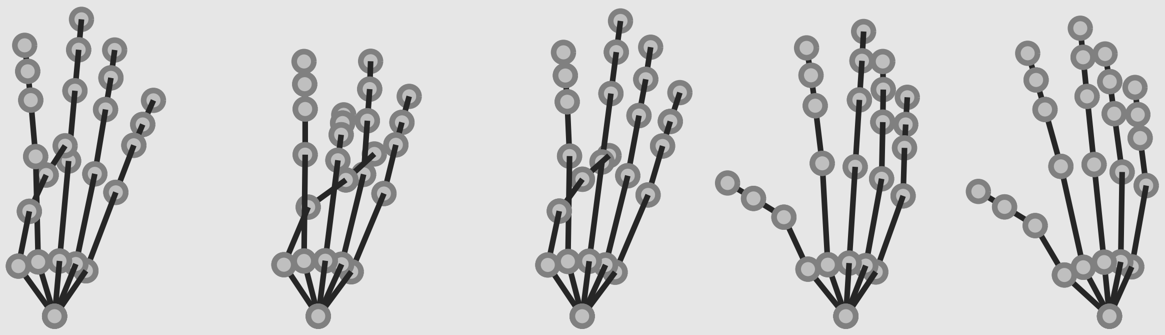


Figure A.3: Objects used for the creation of the database – a pen and a tennis ball as well as an image of all objects for size comparison.



B. Mapping of object and grasp IDs

For convenience, the following table shows the mapping of IDs between grasps, used in *controlled object transport* as described in section 3.6.3, and objects.

Grasp ID	**Object**	**Object ID**
1	Bottle (8 cm diameter)	3
2	Cylinder 25 mm diameter	13
3	Bottle crate	10
4	Cylinder 25 mm diameter	13
6	Cylinder 10 mm diameter	12
7	Cylinder 10 mm diameter	12
8	Cylinder 10 mm diameter	12
9	Business card	8
10	Oval jar (lid)	9
11	Tennis ball (64 mm diameter)	1
13	Tennis ball (64 mm diameter)	1
14	Bottle (cap)	16
15	Bottle crate	10
16	Business card	8
17	Cylinder 25 mm diameter	13
18	Classic notebook (15 mm thick)	11
20	Pen (2 cm diameter)	2
22	Classic notebook (15 mm thick)	11
26	Tennis ball (64 mm diameter)	1
27	Tennis ball (64 mm diameter)	1
28	Tennis ball (64 mm diameter)	1
30	Classic notebook (15 mm thick)	11
31	Glass (62 mm diameter)	5
32	Cylinder 25 mm diameter	13
33	Tennis ball (64 mm diameter)	1

Table B.1: Mapping between grasp IDs, object names, and object IDs.