Seconda Università degli Studi di Napoli

Facoltà di Ingegneria
Dipartimento di Ingegneria dell' Informazione

Tesi di Dottorato in Ingegneria Elettronica

Settore scientifico-disciplinare ING-INF/04
XXIV Ciclo

A kinetostatic data fusion system for observation of human manipulation

Ing. Pietro Falco

Tutor
Prof. Ciro Natale

Coordinatore
Prof. Rocco Pierri

Novembre 2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Description of the work

The robotic systems of the next decade will be, potentially, a part of everyday life as our appliances, servants and assistants, as our helpers and elder-care companions, assisting surgeons in medical operations, intervening in hazardous or life-critical environments for search and rescue operations, and operating in field areas like forestry, agriculture, cleaning, mining, freight transport, construction and demolition, and so on. In this scenario, bringing a robot to the same manipulation skills as those of human beings is recognized as the crucial issue for transferring the robots from industry to the service robotics application domain. Several researchers work towards this objective within the DEXMART project [2]. It attempts to extend a bridge from research on natural cognition to research on artificial cognition, as it will primarily contribute to the development of robotic systems endowed with dexterous and human-aware dual-arm/hand manipulation skills for objects, operating with a high degree of autonomy in unstructured real-world environments. The approach followed in the project to pursue this challenging goal is based on Programming by Demonstration (PbD) strategies, which require development of original methods for interpretation, learning, and modelling from the observation of human manipulation at different levels of abstraction [21]. The adoption of the human observation is becoming more and more frequent within the imitation learning and programming by demonstration approaches (PbD) to robot programming. For robotic systems equipped with anthropomorphic hands, the observation phase is very challenging and no ultimate solution exists. In particular the observation of human manipulation consists of (1) estimation of the hand pose over the time, (2) estimation of joint angles and (3) measurement of fingertip contact forces.

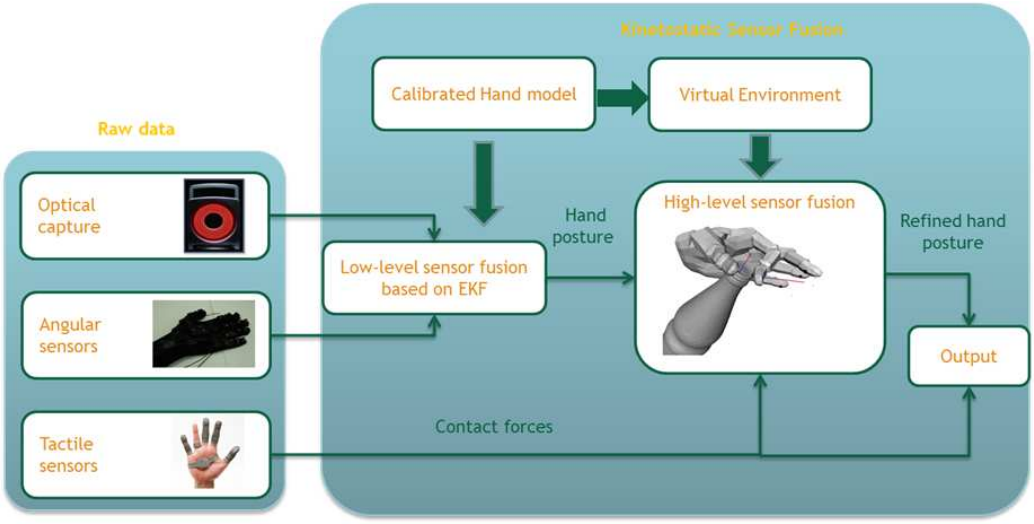


Figure 1.1: Architecture of the proposed sensor fusion system

At the state of the art, the observation of the human hand motion during the execution of complex manipulation tasks is a very difficult problem, handled through two main approaches

- optical position measurements based on motion capture systems, which require many markers and many cameras, for trying to reduce the marker occlusion phenomenon, very frequent in hand tracking problems;
- direct angular measurements, which require complex and expensive sensorized gloves.

In this works, a novel architecture, based on sensor fusion of kinetostatic data, is proposed that overcomes the limitations of the classical systems. As shown in Fig. 1.1, the system is constituted by a low-level sensor fusion module that estimates the hand posture and a high-level module that exploits the knowledge of fingertip contact forces to refine the initial estimation.

The low-level module is in charge to observe human hand motion combining, through a Bayesian senor fusion technique, both the classic approach described above with a significantly decreased hardware complexity, i.e. using a small number of markers (typically only three markers per finger) and cameras, and only three low-cost angular sensors per finger, specifically designed and realized for this purpose. Usually, extracting the joint angles requires a large number of markers because of the high number of human hand DOFs (Degrees of Freedom). Currently, commercial software systems require three

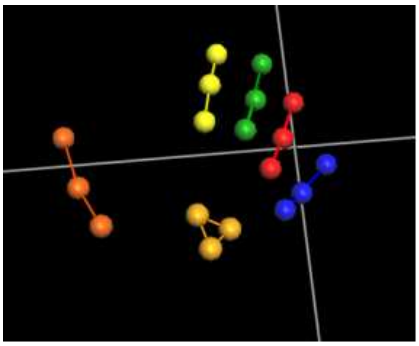


Figure 1.2: Marker set used in the low-level sensor fusion module for the entire hand

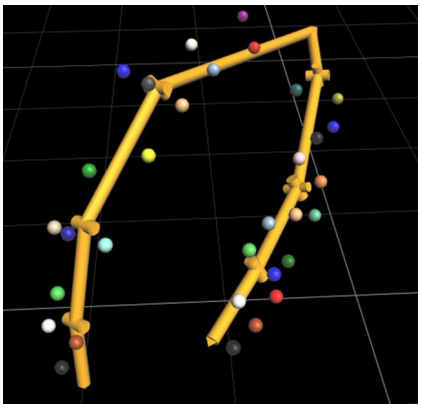


Figure 1.3: Marker set required by commercial software for two fingers

markers per link for correctly estimating joint angle displacements during a human manipulation tasks.

Figures 1.2 and 1.3 show the marker sets required by the designed low-level algorithm and by commercial software for motion capture data processing, respectively. Evidently, the former requires a smaller number of markers and, most importantly, of cameras. This result has been achieved not only by exploiting the additional information coming from the data glove, but also by usefully exploiting the kinematic model of the hand. Basically, the algorithm is constituted by two steps. The first is devoted to estimate the constant kinematic parameters exploiting the recursive nature of the open kinematic chains. The second step consists in estimating the joint angles through a finger-centralized sensor fusion algorithm which takes into account also the marker slipping over the glove surface. The approach followed within the low-level sensor fusion component presents three key innovations:

- design and realization of an optoelectronic low-cost data glove;
- development of a method for estimation of the model parameters exploiting the recursive nature of open kinematic chains;
- development of a method for real-time joint angle tracking though Bayesian sensor fusion algorithms for nonlinear systems.

Concerning the realization of the low-cost data glove, the optoelectronic technology has been selected not only for cheapness sake, but also for its typical interesting properties such as immunity to electromagnetic field, low power consumption and lightness. The data glove proposed is equipped with sensing elements whose developments is based on the use of angle-varying radiation pattern of common LEDs (Light Emitting Diodes) and responsivity pattern of PDs (PhotoDetectors). The effectiveness and advantages of using a solution that exploits this property of optoelectronic components has already been shown in [59], [58] and [11], where the measurement of different physical variables is proposed.

The high-level sensor fusion module aims at improving the observation of the human hand motion, exploiting the measurements of fingertip contact forces and a virtual environment. The main idea of the proposed algorithm is to compare the fingertip contact information, obtained by commercial tactile sensors, with the contact information computed in a virtual environment, that reproduces the real one. In case the estimation of the joint angles and the relative pose between the hand and the object are accurate, the contact information in the virtual and in the real environment are fitting, i.e the contact-consistence condition is satisfied. On the other hand, when the two sources of information are not consistent, a correction of the hand posture is carried out. The correction is constituted by two steps. The first step consists in computing, on the basis of the geometry of the grasped object and of the hand posture, the fingertip position and orientation, such that the contact-consistence condition is satisfied. The second step finds the posture of the hand (i.e. position, orientation and joint angles) such that the end-effectors assume the poses computed in the first step. To tackle this Inverse Kinematics (IK) problem, a Jacobian-based technique known as CLIK (Closed Loop Inverse Kinematics) has been used, which is suitable for the on-line implementation of the correction algorithm ([66], [67], [36]). Since the starting point of the IK algorithm is given by the data from the sensors, only a local correction is required and Jacobian based methods, that are fast to find local minima, are particulary suitable to this application. It is important to emphasize that in the inverse kinematics algorithm, the hand is modeled not as five independent kinematic chains, but as a "kinematic tree"

with a root and five branches. The root is composed by six 1DoF joints, describing the pose of the hand in space and the branches are the five serial chains describing the five fingers. The correction brings two advantages: improve of the accuracy of the hand observation and guarantee of the coherence between hand posture and measured force in the virtual environment, which is very important for PbD applications.

In order to correctly tune the parameters of the CLIK algorithm, a study on the stability of closed loop inverse kinematic algorithms has been carried out. It provides a convergence analysis of classical inverse kinematics algorithms for redundant robots, whose stability is usually proved only in the continuous-time domain, thus neglecting limits of the actual implementation in the discrete-time. Whereas, the convergence analysis carried out in this work in the discrete-time domain provides a method to find bounds on the gain of the closed-loop inverse kinematics algorithms in relation to the sampling time. It also provides an estimation of the region of attraction (without resorting to Lyapunov arguments) i.e. upper bounds on the initial task space error.

A further result is the development of a novel robot calibration procedure that uses a motion capture system. This methods exploits the results already obtained in the hand calibration procedure. The novelty of the procedure consists in the adoption of an optical motion capture system that allows a 3D position measurement distributed along the entire kinematic chain, not only on its end effector. This enables the estimation of the kinematic parameters in a closed form without the need to resort to any linearisation of the error function. The robustness of the estimation algorithm against measurement noise is guaranteed by the adoption of multivariate statistical methods like the Principal Component Analysis (PCA). The procedure is intended for robot manufactures, who can adopt it to select the estimated kinematic parameters, to be used in the control units of the robot for the computation of both direct and inverse kinematics, for improving the absolute positioning accuracy of the end effector. Both simulation and experimental results obtained in the calibration of an actual industrial 6DOF robot are reported, which confirm the effectiveness of the proposed approach.

1.2 State of the art

1.2.1 Observation of human manipulation

Until now, only few papers have addressed the problems of kinematic model parameter and joint angle estimation in human complex manipulation tasks;

none of these papers uses approaches based on the sensor fusion. In [53] a protocol to determine the link structure of human hand using motion capture data is proposed. [75] describes a global optimization method for off-line assessing joint angles in human hand and for calibrating instrumented glove from motion capture system measurements. The papers [12] and [79] use, respectively, a deterministic and a stochastic global optimization algorithm to determine the centers and the axes of rotation for fingers in a simplified model of the human hand. In [16] an anatomic-based cost metric is proposed to identify a model of the carpometacarpal joint of the thumb, that is suitable for measuring mobility. [45] and [13] try to estimate the whole body motion from the measurement by a Kalman-like approach, but without focusing on the hand motion. [33] is one of the first works that address the real-time finger tracking problem; it uses a Kalman approach to track the marker positions for an augmented reality application. In PbD, visual observation of fingertip motion has been investigated, e.g. for simple grasps [74]. In general, complex dexterous manipulation tasks can not be fully observed due to occlusions of the fingers during manipulation, e.g. in the Visualeyez system [57]. Additionally, magnetic field trackers can be used to track the fingertip motion [34] but limit the movability of the human operator. In contrast, datagloves measure the finger joints directly but are quite noisy and require a calibration in each demonstration. In [75], a 14DOF dataglove is calibrated using an infrared tracking system. In [5] a data glove based on goniometric sensors is described, that is less complex and less expensive than the most popular commercial ones.

In [33] and [75] methods to track the hand movement in the space are proposed, which elaborate measurements from optical motion capture systems. Yet, these kind of approaches require many markers and many cameras, due to the marker occlusion phenomenon and the high number of human hand DOFs concentrated in a small volume.

Until now, tactile sensors have been used to improve the segmentation of the sensor data into elementary actions [82] but no qualitative information about contact points has been used to improve the observation result. In [37], sixteen values for the phalanxes and abductions and two force measurements for the thumb and middle finger are measured and a neural network is trained to reproduce the grasp but the object geometry and the inconsistency of joint measurements and force measurements is not taken into account.

1.2.2 Inverse kinematics

Inverse kinematics of robotic systems is one of the basic problems in robotics. It has been addressed and solved in a variety of manners both for non-

redundant and redundant robots. At the beginning it was addressed by resorting to classical numerical methods, such as Newton-Raphson algorithm for finding zeros of nonlinear functions [10], [39], or to more general optimization algorithms [80]. Later on it has been solved by inverting differential kinematics in a closed-loop fashion by viewing the inverse kinematics problem as a feedback control problem [65, 27], leading to the so-called CLIK (Closed-Loop Inverse Kinematics) algorithm. More recently, mixed numerical-analytical approaches have been proposed leading to approximate solutions [71]. One of the main problems related to inverse kinematics is the handling of multiple tasks. Task-priority redundancy resolution techniques [51, 55] were proposed to allow the specification of a primary task that is fulfilled with higher priority with respect to a secondary task. The same objective has been reached by resorting to a null-space based solution in [19], whose extension to any number of tasks has been recently presented in [6], still in the framework of CLIK algorithms. For a comprehensive review, the reader is referred to [76] and references therein.

The main difficulty in the study of algorithmic solutions to the inverse kinematics problem, is related to the discrete-time nature of the dynamic system at hand, combined with its strong nonlinearity, deriving from the nonlinearity of the kinematics. Very few papers cope with this problem, e.g. [27],[28] proposed Lyapunov-based arguments to prove stability.

1.3 Benefits and limitation of data fusion

1.3.1 Benefits

The benefits of sensor fusion are both qualitative and quantitative. Qualitative benefits are improved operational performance, extended spatial coverage, extended temporal coverage, increased higher probability of correct inference, reduced ambiguity of inference, improved detection, enhanced spatial resolution, improved system reliability, increased dimensionality [52].

- Robust operational performance: one sensor can contribute information while other are unavailable, jammed, or lack coverage of a target or event.
- Extended spatial coverage: one sensor can look where another cannot
- Extended temporal coverage: one sensor can detect/measure an event/target when another cannot Probability of detection increased

- Increased confidence: one or more sensors can confirm the same target or event
- Reduced ambiguity: joint information from multiple sensors reduces the set of hypotheses about the target or event
- Enhanced spatial resolution: multiple sensors can geometrically form a synthetic aperture capable of greater resolution than a single sensor can form.
- Increased dimensionality: a system employing different sensors to observe different physical phenomena are less vulnerable to disruption.
- Improved system reliability: multiple sensor suites have an inherent redundancy.

A quantitative aim of sensor fusion is to improve the accuracy of the observation, e.g. a GPS position determination. An attempt to illustrate the quantitative benefits of sensor fusion is described in [54]. In the article, data from N identical sensors are fused to classify an observed phenomenon following a majority vote rule. The sensors are assumed to be statistically independent and the a priori probabilities are taken equal to $\frac{1}{N}$. Figure 1.4 plots the increased probability of correct inference when the number of sensors is increased from 5 to 7 (the bottom curve of Fig. 1.4), from 3 to 5 sensors (middle curve) and from 1 to 3 sensors (top curve).

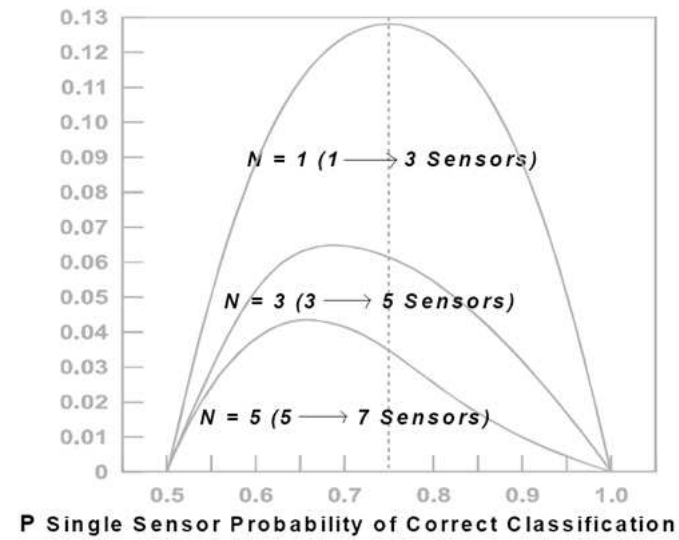


Figure 1.4: Marginal gain in correct classification with addition sensors

The results in [54] provide some conceptual rule of thumb:

- Combining data from multiple sensors which have probability of correct detection less than 0.5 does not provide significant advantages.
- Combining data from multiple sensors that have very high probability of correct detection, e.g. around 0.95 does not provide significant advantages
- When N is very large (e.g. greater than 10 sensors) ,adding new identical sensors does not provide advantages. Adding sensors of other kinds,a substantial increase of information content may be expected, depending on fusion objective.
- The greatest marginal improvement in sensor fusion occurs for a moderate number of sensors (i.e. 1 to 7), each having a reasonable probability of correct identification.

1.3.2 Limitations

In the design of a sensor fusion system is of major importance to consider the limitations of data fusion; in [25], Hall and Steinberg describe the "dirty secrets" of data fusion:

- There is no substitute for a good sensor: no amount of data fusion can replace the data of a single accurate sensor. For example in a mechanical system, conditions such as operating speed and vibration level can give useful information about system operation but they do not provide estimate of other quantities such as the system operating temperature
- Downstream processing cannot make up for errors or failures in upstream processing: data fusion process cannot correct errors in preprocessing of individual sensor processing.
- Sensor Fusion can result in poor performance if incorrect information about performance is used: a common failure of data fusion is to characterize the sensor performance in an ad hoc or convenient way; doing so, the sensor weight will be set by the fusion algorithms in incorrect manner.
- There is no such thing as a magic or golden data fusion algorithm: there is no algorithm optimal in all conditions and often real applications do not meet the underlying assumptions required by data fusion algorithms.

- There will never be enough training data: the needed training data for pattern recognition algorithm are never sufficient, so hybrid methods such as model-based methods, syntactical representation, combination of methods are necessary.

- It is difficult to quantify the value of a data fusion system: a challenge in data fusion systems is to quantify the utility of system at mission level, i.e. to understand how the data fusion system performs well in operational environment.

- Fusion is not a static process: The Data Fusion process is not static but it is an iterative process which continually seeks to improve the estimates of a situation observed or a threat environment.

1.4 Bayesian sensor fusion

A nonlinear stochastic system can be defined by a stochastic discrete-time state space transition equation:

$$\boldsymbol{x}_k = \boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) \tag{1.1}$$

and the stochastic observation process:

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{1.2}$$

where, at time k, $\boldsymbol{x}_k$ is the system state vector, $\boldsymbol{w}_k$ is the dynamic noise vector, $\boldsymbol{y}_k$ is the observation vector and $\boldsymbol{v}_k$ is the observation noise vector. The deterministic functions $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ link the prior state to the current state and the current state to the observation vector, respectively.

When $\boldsymbol{y}_k$ is the output vector of a redundant number of sensors, a Bayesian filtering problem becomes a sensor fusion problem. In a Bayesian context, the problem is to quantify the posterior density $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ where the observations are specified by $\boldsymbol{y}_{1:k} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k\}$. The above nonlinear non-Gaussian state-space model, eq. 1.1, specifies the predictive conditional transition density, $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{1:k-1})$, of the current state given the previous state and all previous observations. Also, the observation process equation, 1.2, specifies the likelihood function of the current observation given the current state, $p(y_k|x_k)$. We suppose that the Markov assumption holds, such as:

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{y}_{1:k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{1.3}$$

The prior probability, $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$, is defined by Bayes rule as:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})d\boldsymbol{x}_{k-1} \tag{1.4}$$

Here, the previous posterior density is identified as $\boldsymbol{p}(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ eq. 1.4 is called prediction step. The correction (or update) step generates the posterior probability density function from:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = cp(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) \tag{1.5}$$

where c is a normalization constant.

The MMSE Bayesian filtering problem is to estimate, in a recursive manner, the first two moments of pdf $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$, using eq. 1.4 and eq. 1.5. That is, the estimated state at the time k, $\hat{\boldsymbol{x}}_k$ will be:

$$\hat{\boldsymbol{x}}_k = E[\boldsymbol{x}_k|\boldsymbol{y}_{1:k}] = \int \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})d\boldsymbol{x}_k \tag{1.6}$$

But for a general multivariate distribution the integrals 1.4 and 1.6 cannot be evaluated in closed form, so some form of integration approximation must be made. In particular we obtain:

1. Kalman Filter (KF) when $\boldsymbol{f}_k$, $\boldsymbol{h}_k$ are linear function and $\boldsymbol{v}_k$, $\boldsymbol{w}_k$ are AGWN stochastic variables

2. Extended Kalman Filter when $\boldsymbol{f}_k$, $\boldsymbol{h}_k$ are nonlinear function but are approximated with their first order Taylor series and $\boldsymbol{v}_k$, $\boldsymbol{w}_k$ are AGWN stochastic variables

3. Unscented Kalman Filter when the Unscented Trasform is used in order to propagate the AGWN variables $\boldsymbol{v}_k$, $\boldsymbol{w}_k$ through the deterministic nonlinear functions $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$.

4. Particle Filters when Monte-Carlo-like methods are adopted to resolve the integrals 1.4 and 1.6, it does not need any restrictive hypothesis on the pdf of stochastic variables $\boldsymbol{v}_k$, $\boldsymbol{w}_k$.

In [73] and [63] it is possible to find a complete explanation on the analytical and numerical approximations of the Bayesian filter.

Chapter 2

Sensing system

2.1 The sensory environment

The measurement system adopted for tracking human hand motions uses two different types of sensing systems. The first one is a commercial optical motion capture system (see Fig. 2.1). The second one is a sensorized glove equipped with three angular sensors per finger and with three reflective markers per finger. Marker positions are measured through a Vicon optical motion capture system, composed by four CMOS cameras, a workstation and a host PC, on which a real-time engine and the software for calibration and management are installed. The workstation is connected to a PC through an Ethernet cable, with TCP/IP protocol. The data glove has been principally

Figure 2.1: Cameras of the Vicon mocap system

designed to be used in a sensor fusion system and the concept is presented in Section 2.2. The calibration curve of the sensor can be estimated according

to the specific procedure described in Section 3.1.2. Both the motion capture system and the data glove work with a sampling rate of 60 Hz. The details of the approach followed to synchronize the two systems and to avoid interference of the optoelectronic sensors with the infrared cameras are reported in Section 2.2.3. In order to measure the fingertip contact forces, commercial tactile sensors have been placed under the data glove see Fig. 4.3. In each fingertip, a tactile sensor pad is used to measure the force intensity applied to an object. Due to the size and location of the pads, the human operator has to consider the sensor pads and adapt the manipulation motion in order to make consistent force measurements. An additional palm sensor is available but has not been used in this work. The system is calibrated using a provided dynamometer. The repeatability is $< 4\%$ of the full scale range, which is 4.55 to 22.73kg [4].

2.2 Data glove design

2.2.1 Sensing element concept

In the data glove design process, the rotational center of each single joint of the human hand has been considered fixed in the whole angular range of the joint (about $[0°, 90°]$). Therefore, the kinematic behavior of a single joint can be modelled with good approximation as an ideal 1-DOF revolute joint. As a consequence, it is possible to introduce a simplified geometrical model of two adjacent phalanges, as illustrated in Fig. 2.2, and define the joint angle α. The rotational center of the joint is set at the origin O of a Cartesian coordinate system, in which xOz constitutes the plane of rotation of the phalanges. The proposed angular sensor takes advantage of the

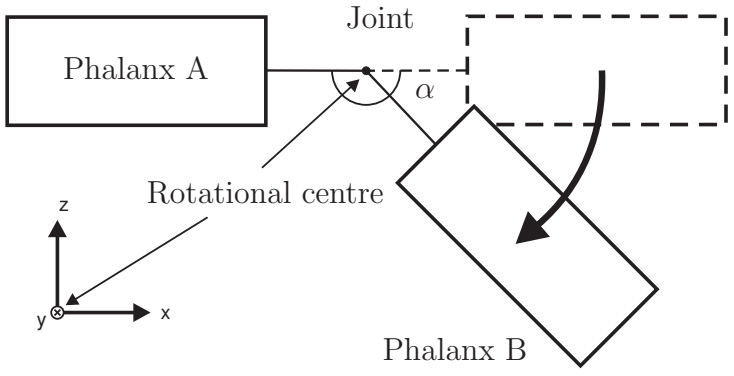


Figure 2.2: Schematics of the two-phalanges simplified mechanical model.

angle-varying radiation pattern of common LEDs and responsivity pattern of photodetectors. Consider a single joint of a finger in the rest position ($\alpha = 0°$). A LED and a photodetector are placed respectively on the first and second phalanx attached to the joint, facing each other with mechanical axes overlapping. In this state a certain amount of light emitted by the LED reaches the photodetector and it is proportionally converted into an electrical current, I_0. As the joint starts to flex ($\alpha > 0°$), the mechanical axes of the emitter and of the receiver experience some angular displacement. In this new condition a different amount of light will be sensed by the photodetector and converted into a current different from I_0. This happens because the radiation pattern of the LED varies with the observation angle, so that the receiver detects different values of radiant flux in the two cases. At the same time also the way the photodetector weighs received light varies, as its relative position and orientation respect to the source change, according to its responsivity pattern. The combination of these two effects leads to the observed variations of the photocurrent.

Being the electrical current of the photodetector variable with the flexure angle of the joint, it is possible to introduce the function $I(\alpha)$. If $I(\alpha)$ is a monotonic function of its argument, a measure of its value is uniquely associated to a value of the angle α. Thus if the power of light emitted by the LED is kept constant and the photodetector current is measured, the flexure angle of the joint at every instant can be reconstructed. A LED/photodiode couple is needed for each joint whose angular displacement has to be detected. The diagram in Fig. 2.3 recalls the simplified model of a single finger joint presented in Fig. 2.2, reporting the unique joint angle α and $\beta = \alpha/2$, that will be conveniently used in the following. Figure 2.3 shows that the LED and the photodetector have a fixed displacement from the axis of the phalanx to which they are attached. The position of the emitter and the detector can be specified with respect to the fixed Cartesian coordinate system $Oxyz$. The origin O of the system is chosen to coincide with the center of rotation of the joint, the x-axis overlapping the axes of the phalanges when $\alpha = 0°$. Assuming the devices to lie symmetrically with respect to the z-axis, their angle-dependent coordinates are (p_x, p_z) for the photodetector and $(-p_x, p_z)$ for the LED. Their variation with α has been subtended. When the joint is in its rest position ($\alpha = 0°$), it is possible to define $p_x = p_{x0}$ and $p_z = p_{z0}$. As α varies both p_x and p_z vary, together with the distance d between the tips of the LED and photodetector. In particular

$$d(\beta) = 2p_x(\beta) = 2\left[p_{x0}\cos(\beta) + p_{z0}\sin(\beta)\right], \tag{2.1}$$

which for $\beta = \alpha = 0°$ gives the initial distance between the devices $d = d_0 = 2p_{x0}$. The initial positioning of the devices (in terms of p_{x0} and p_{z0}) repre-

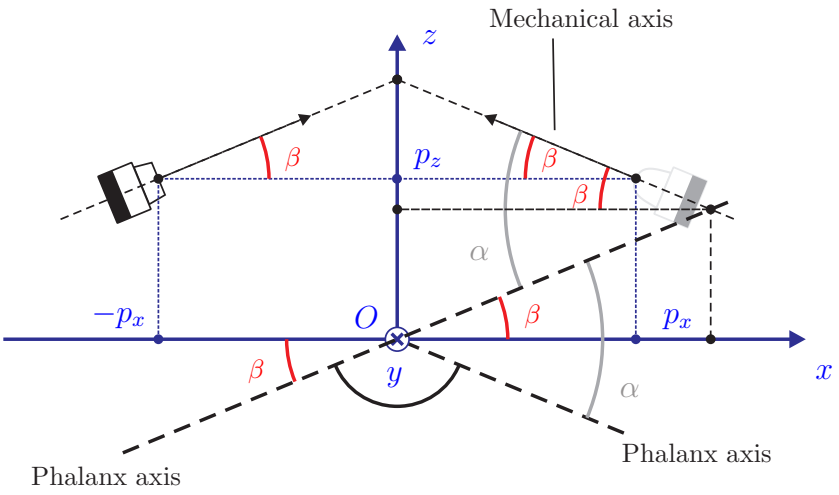


Figure 2.3: Relationship between angles in the geometrical model.

sents a degree of freedom that can be used to alter the d-β characteristic and in particular its monotonicity, consequently changing the sensitivity of the sensor. Owing to the symmetry of the considered system, with elementary geometric considerations on Fig. 2.3, when the joint exhibits a certain bending angle α, the mechanical axes of both the LED and the photodetector form an angle β with the x axis. The symmetric positioning of the components with respect to the center of rotation of the joint is not a mandatory condition to have a well working sensor. The choice made here only simplifies the geometric considerations used to optimize the positioning and the calibration of the sensor. In general the couple LED/photodetector can be used as an angular displacement sensor also without a symmetric positioning.

At this point, recalling the theory on LED radiation patterns [46], it is possible to look for a model of the system in order to optimize the design of the sensor, selecting the initial positioning (p_{x0} and p_{z0}) of the devices. Fig. 2.4 shows what happens in terms of radiated and received radiant flux in the two different conditions $\alpha = 0°$ and $\alpha \neq 0°$. In the picture two new variables are introduced, θ_T representing the angle between the LED mechanical axis and the segment from the tip of the photodetector to the tip of the LED, and θ_R representing the angle between the photodetector mechanical axis and the same segment. Both quantities are positively measured clockwise. According to what was shown above, with reference to Fig. 2.3, θ_T is equal to β and θ_R to $-\beta$. If the distance d was large enough to render the far-field approximation valid, the LED could be regarded as a point source. In that case the photocurrent $I(\cdot)$ (and thus the received radiant flux by the photodetector) will be proportional to the product between the

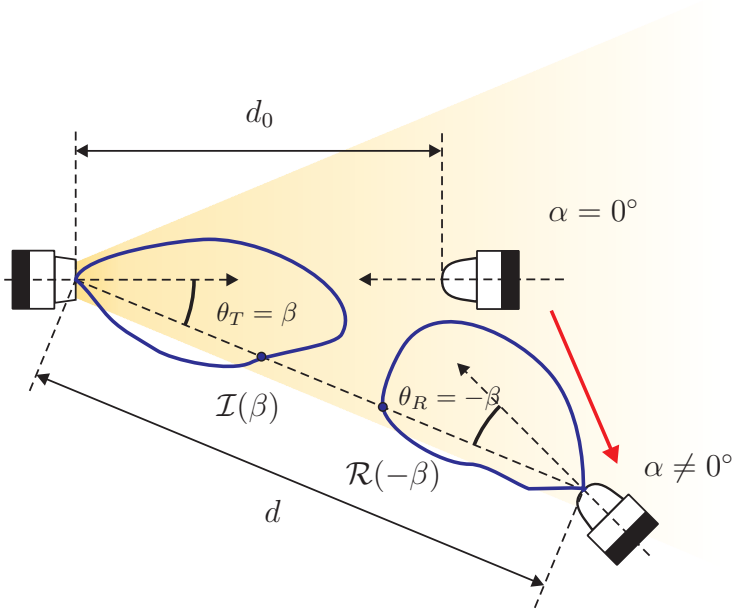


Figure 2.4: Mutual orientation of the LED source and the photodetector.

radiant intensity pattern of the LED, $\mathcal{I}(\theta_T)$, evaluated in β and the responsivity pattern of the photodetector, $\mathcal{R}(\theta_R)$[1], evaluated in $-\beta$, and inversely proportional to the squared distance $d(\beta)$

$$I(\alpha/2) = I(\beta) = K\frac{\mathcal{I}(\beta)\mathcal{R}(-\beta)}{d^2(\beta)}, \tag{2.2}$$

where K is a dimensional multiplicative constant. It is evident from 2.2 that the sensor characteristics, in terms of sensitivity and measurements range, depend on LED radiant intensity pattern, photodetector responsivity pattern and relative positioning variations of the components. The two patterns are available on datasheets of the selected components, while positioning variations can be derived by simple geometrical considerations.

2.2.2 Sensing element realization

A preliminary testing phase has been performed to select the optoelectronic components among various commercially available devices. The selection has been made taking into account the nominal beam angle of the LED and the acceptance angle of the photodiode so as to guarantee an acceptable sensitivity in the whole angular range of interest. The selected devices used

[1]the φ-dependence of the radiation and responsivity patterns is here omitted since the devices only move within a plane at constant φ.

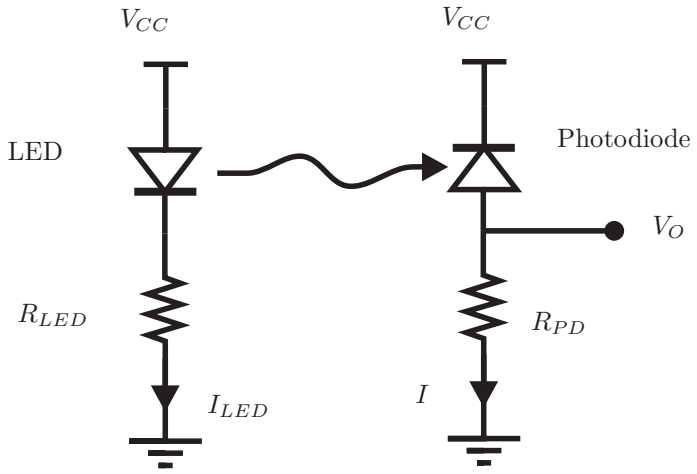


Figure 2.5: Schematics of the emitter and receiver circuits for the second sensor prototype.

for experimental implementation are branded Avago Technologies Inc. and are spectrally matched with an infrared peak wavelength of 875 nm. Such a choice in terms of wavelength range guarantees a sufficient robustness against ambient light. The emitter (manufacturer code number HSDL-4400) is an AlGaAs flat-top light emitting diode featuring a nominal beam angle of 110°, an on-axis radiant intensity of 6 mW/sr corresponding to a 100 mA maximum forward current, and a bandwidth of about 7 MHz. The detector (manufacturer code number HSDL-5420) is a domed PIN photodiode characterized by a 28° acceptance angle and a nominal optical bandwidth of 50 MHz. The emitter and receiver circuit schematics are depicted in Fig. 2.5, where it is evident that the photocurrent is simply transformed into a voltage, without any conditioning electronics for signal amplification. An additional electronics can be used to improve the robustness with respect to noise and disturbances.

In order to fix the initial positioning of the components a mold resin has been prepared and used to achieve a form of elastic material with the two devices integrated within it, as shown in Fig. 2.6. The elastic layer has been realized using a two component room temperature condensation curing silicone compound (MM906 produced by ACC Silicones). The cured silicone is an exceptionally flexible rubber with very high mechanical properties and good shelf life stability. This integration guarantees that the two devices are facing each other with mechanical axes overlapped, when no angular displacement is applied to the sensing element. The selection of a material with an high elasticity and a low hysteresis is fundamental to obtain a sensing

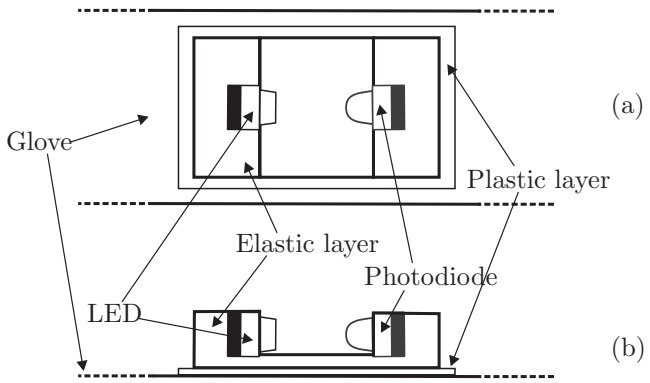


Figure 2.6: Drawings of a sensing element: top view (a) and lateral view (b).

Figure 2.7: Picture of a single sensing element.

element with a good measurement repeatability. After the realization of a sensing element, another important point has been the bonding of the sensing element to the neoprene glove. The elasticity of the glove implies that the angular variation of a single joint of the human hand produces the strain of the glove also in areas different from than directly affected by the angular motion. Clearly, the elastic layer with the two devices integrated within it, if directly bonded to the glove, is sensitive to this strain. Therefore a thin plastic layer has to be inserted between the elastic layer and the glove as shown in Fig. 2.6. This thin plastic layer, for each joint, allows the transfer from the human hand to the elastic layer of the angular displacement but not of the strain. The use of a sensing element constituted by an elastic layer and a thin plastic layer avoids coupling problems caused by glove elasticity for measurements of neighboring joints. The Fig. 2.7 shown a picture of a realized sensing element bonded to the glove.

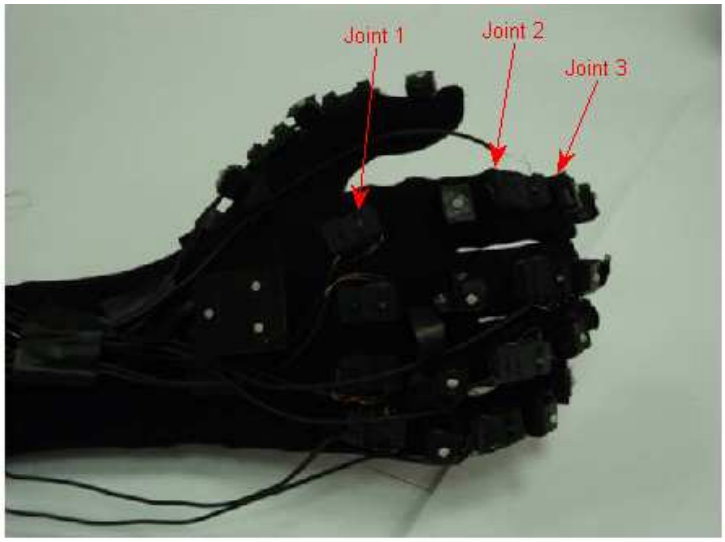


Figure 2.8: Picture of the data glove prototype with angular sensors and markers used for motion capture system.

2.2.3 The sensorized glove

A prototype of a data glove has been realized installing sensing elements on the joints of a commercial neoprene glove. The Fig. 2.8 shows a picture of the realized prototype. Figure 2.9 reports typical measured voltages for all index finger joints in a simple pick and place task. In order to calibrate the data glove, a recursive inverse kinematic algorithm has been used. It elaborates measurements from a motion capture system and exploits the recursive nature of the open kinematic chains to calculate the joint angles. From the user point of view, a calibration session consists of observing, by an optical motion capture system, repeated flexion-extension motion of all the fingers. It is important to underline that the result of the calibration procedure depends on the shape and dimension of the user's hand. Then, for a different performer, a new calibration procedure is required. A first, but not accurate enough, version of the calibration algorithm can be found in [64].

Figures 2.10, 2.11, 2.12 report examples of calibration curves for the index finger joints of the data glove prototype, , which have been obtained through the calibration procedure described in Section 3.1.2. Obviously, the range of angular motion is different for each joint and also the sensitivity. It is important to underline that the voltages reported in Figures. 2.10, 2.11, 2.12 have been obtained using the voltages V_0 as reported in Fig. 2.5, without any conditioning electronics for signal amplification.

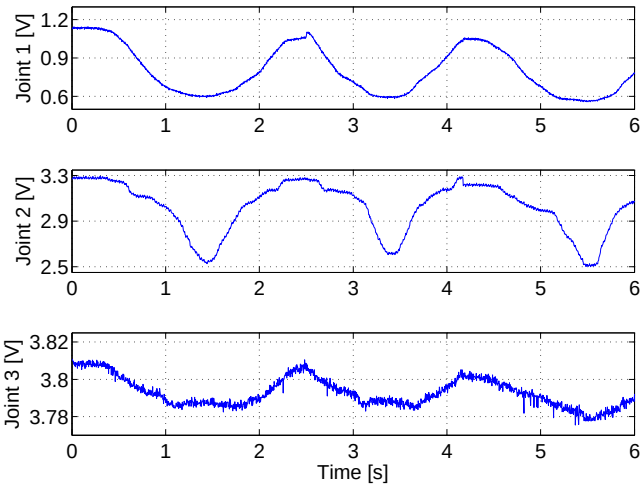


Figure 2.9: Typical measured voltages on joints pointed out in Fig. 2.8 for a pick and place task.

2.2.4 Synchronization with cameras

The combined use of the proposed joint angle sensors and the motion capture system needs the design of a suitable conditioning electronics in order to synchronize the data captured by the two different systems. Since the presented optical sensors work in the same wavelength range of the motion capture system, the conditioning electronics must also handle the problem of optical interference between the two measuring systems: the PDs used for the angular sensors realization are sensitive to strobes of the cameras and as a consequence the measurements of the joint angles may be wrong if the sampling occurs while the strobes are turned on; the CMOS cameras of the motion capture system are sensitive to light emitted by LEDs and so the LEDs can be confused with the markers. The optical interference and synchronization problems have been tackled and solved with the same conditioning electronics. The basic idea is to alternate the operation of two systems, so that the angular sensors are switched off when the cameras capture the markers and the angle measurements occur when the camera strobes are switched off. The scheme of the conditioning electronics is reported in Fig. 2.14 and the corresponding timing of the procedure is reported in Fig. 2.14. The input signal from the motion capture system is constituted by impulses of known length Δ with a frequency $f = 1/T$, where f is the working frequency of the capture system (typically $f = 60$Hz). The marker positions are captured with a frequency f for each instant t_0 that coincides

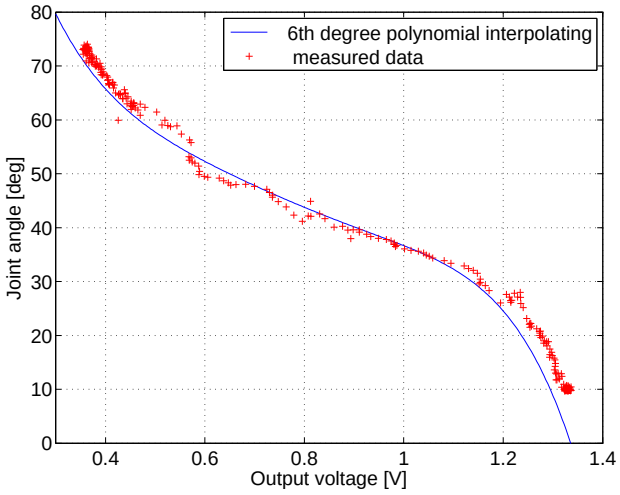


Figure 2.10: Calibration curves of joint 1 for a single finger as reported in Fig. 2.8.

with the center of the time interval Δ. The camera strobes are switched on for a known time $\tau > \Delta$, centered with respect to the t_0 instants, in order to ensure the optimal illumination during the capture. The microcontroller, on the basis of the input signal, generates the driving signal for angular sensors which is low when the strobes are switched on and is high otherwise. This signal is used as reference for the power amplifier that provides power supply for LEDs and PDs. Using this driving signal, the optical interference between the two systems is avoided. The microcontroller also manages via SPI (Serial Peripheral Interface) an A/D converter where the output signals of PDs are connected. The microcontroller captures the PD signals at sampling instant t_s, delayed with respect to t_0, so that LEDs and PDs are working and strobes are switched off. Since each instant t_s is calculated with respect to the previous impulse Δ, generated by the motion capture system, the synchronization of the angular sensor data is automatically satisfied with a sampling frequency f.

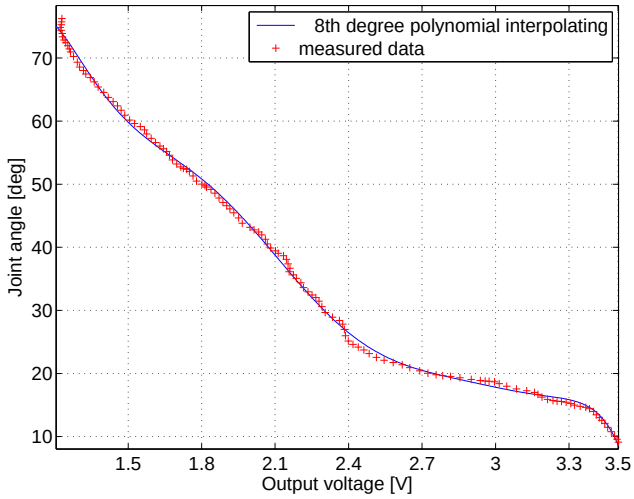


Figure 2.11: Calibration curves of joint 2 for a single finger as reported in Fig. 2.8.

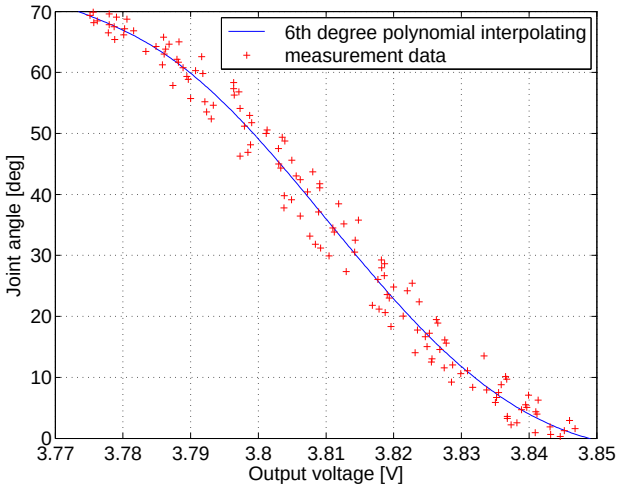


Figure 2.12: Calibration curves of joint 3 for a single finger as reported in Fig. 2.8.

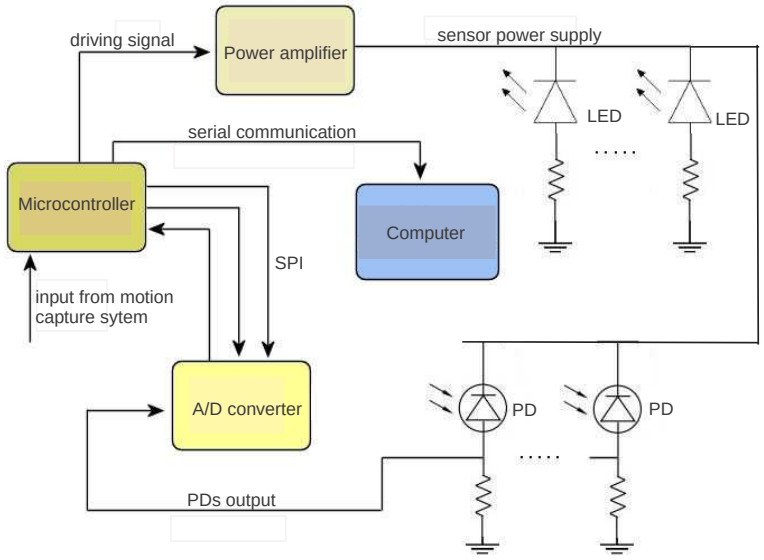


Figure 2.13: Working scheme of the conditioning electronics

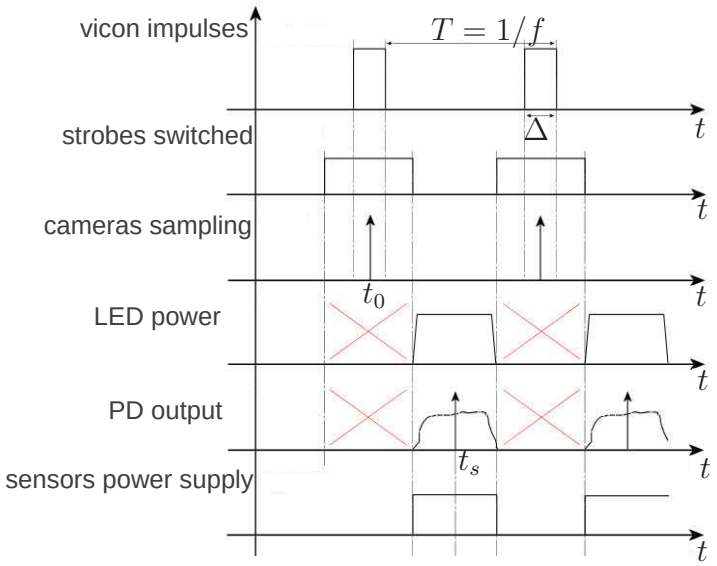


Figure 2.14: Timing of the conditioning electronics signals.

Chapter 3

Low-level sensor fusion

The low-level sensor fusion module is in charge of estimating the hand posture through the fusion of angular sensor voltages and marker positions from the motion capture system. In order to properly work, it requires the calibration of the human hand model and the calibration of the angular sensors placed on the sensorized glove. The calibrated hand model is also exploited within the high-level sensor fusion module, described in Chapter 4.

3.1 Hand model and calibration

In order to reconstruct the motion of the human hand, a kinematic model has to be selected. Then, its parameters have to be estimated, including the joint angles necessary to animate the model.

3.1.1 Model definition

A universally recognized as an accurate model of the human hand is the one proposed in [60], which allows to describe also palm arching movements. A simplified version of the model is adopted in this work assuming a rigid palm. In detail, for each finger, the selected 4−DoF kinematic model is depicted in Fig. 3.1, where also the markers attached to each bone are reported. The above kinematic model assumes that the first two joints of each finger are two consecutive pin joints with orthogonal axes. Another key assumption of the adopted kinematic model is that the flexion axes of each finger are all aligned. The method adopted to define the kinematic model, i.e. the relationship between the joint angles and the fingertip poses, is the so-called Denavit-Hartenberg technique [29], which is widely used both in robotics and biomechanics. The reader unfamiliar with this method can refer to,

link	a	d	ϑ	α
1	0	0	ϑ_1	$\frac{\pi}{2}$
2	a_2	0	ϑ_2	0
3	a_3	0	ϑ_3	0
4	a_4	0	ϑ_4	0

Table 3.1: D-H table for all the fingers.

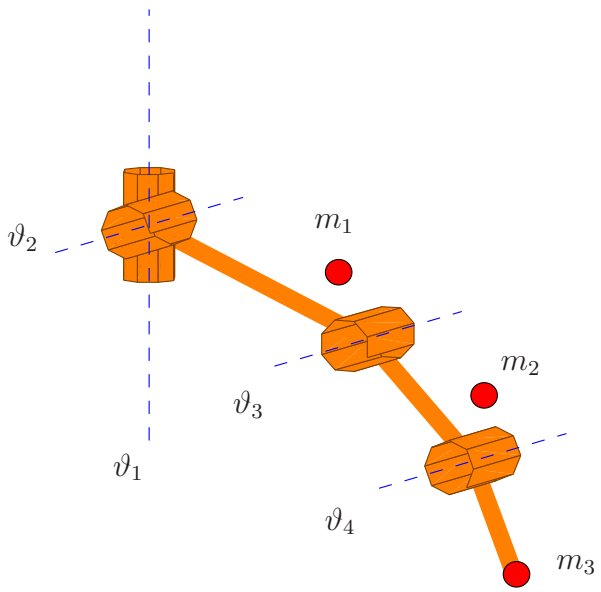


Figure 3.1: Kinematic model and marker set for one finger

e.g., [67]. The resulting Denavit-Hartenberg (D-H) parameter table is the one in Tab. 3.5. The D-H reference frames fixed to the links of the finger are depicted in Fig. 3.2, together with two intermediate frames used in the calibration algorithm.

3.1.2 Model and angular sensor calibration

In order to apply the Bayesian sensor fusion technique, two calibration steps are needed

1. calibration of the hand model
2. calibration of angular sensors

The calibration of the hand model consists of calculating, for each finger, the D-H parameter a_2, a_3, a_4 and the pose of the D-H frame 0 with respect to the palm-fixed frame w.

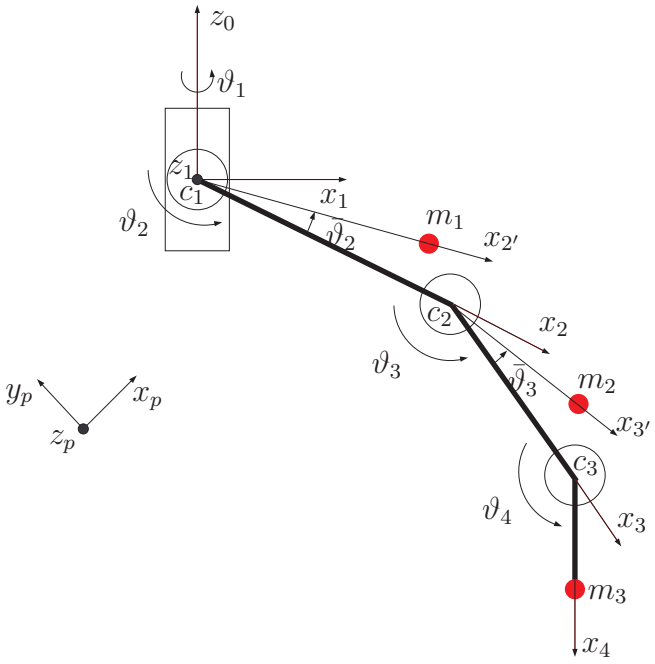


Figure 3.2: D-H frames and markers attached to the finger

The calibration of an angular sensor consists of estimating the relation between the sensor output voltage v and the joint angle ϑ. Due to mounting constraints, the angular sensors can be installed only on the joints 2, 3, 4 of each finger. For both hand model and angular sensor calibration, the motion capture system is used. In a special calibration session, the user has to perform repeated motions of all the fingers trying to avoid abduction/adduction movements and to minimize marker occlusions. This ensures that the three markers on each finger move on parallel planes, thus allowing robust estimation of the parallel joint axes. The estimation of parallel joint axes is performed by the Principal Component Analysis (PCA).

In general, the central idea of PCA is to transform a p-dimensional data set D_p into a m-dimensional data set D_m, with $m < p$, losing as little information as possible.

Suppose that $\boldsymbol{x}$ is a vector of p random variables. The variance and covariance of $\boldsymbol{x}$ are of interest. More in details, the problem of PCA consists of looking for a few ($\ll p$) derived variables that preserve most of the information given by variance and covariance of $\boldsymbol{x}$. The first step is to look for a linear function $\boldsymbol{a}_1^T\boldsymbol{x}$ of the elements of $\boldsymbol{x}$ having maximum variance, where $\boldsymbol{a}_i$ is a vector of p constants a_{11}, a_{12}, ..., a_{1p}. The next step is to look for a linear

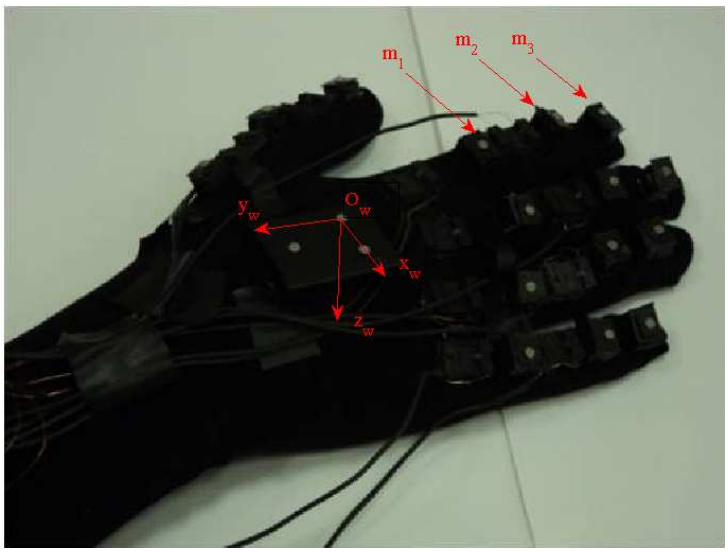


Figure 3.3: Data glove prototype

function $\boldsymbol{a}_2^T\boldsymbol{x}$ uncorrelated with $\boldsymbol{a}_1^T\boldsymbol{x}$ having maximum variance, and so on, so that at the k-th stage a linear function $\boldsymbol{a}_k^T\boldsymbol{x}$ is found that has maximum variance subject to being uncorrelated with $\boldsymbol{a}_1^T\boldsymbol{x}$, $\boldsymbol{a}_2^T\boldsymbol{x}$, ..., $\boldsymbol{a}_{k-1}^T\boldsymbol{x}$. The k-th derived variable, $\boldsymbol{a}_k^T\boldsymbol{x}$ is the k-th Principal Component (PC). Up to p PCs could be found, but it is hoped, in general, that most of the variation in x will be accounted for by m PCs, where $m \ll p$. If so, the data set D_p can be well approximated with the new data set D_m. In fact, if a set of p variables has substantial correlations among them, then the first few PCs will account for most of the variation in the original variables. Conversely, the last few PCs identify directions in which there is very little variation; that is, they identify near constant linear relationship among the original variables. PCA is theoretically the optimum transform for a given data in least square terms. PCA involves the calculation of the eigenvalue decomposition of a data covariance matrix or singular value decomposition of a data matrix, usually after mean centering the data for each attribute. For a data matrix, $\boldsymbol{X}$, with zero empirical mean (the empirical mean of the distribution has been subtracted from the data set), where each row represents a different repetition of the experiment, and each column gives the results from a particular probe, the PCA transformation is given by:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}$$

where $\boldsymbol{V}^{-1}\boldsymbol{U}\boldsymbol{\Sigma}$ is the singular value decomposition of $\boldsymbol{X}$. For a complete and detailed coverage on Principal Component Analysis, [44] is a good reference.

Finger	a_2 [mm]	a_3 [mm]	a_4 [mm]
Thumb	49.13	31.79	24.11
Index	45.02	25.53	20.47
Medium	50.07	29.64	19.16
Ring	48.40	22.42	20.92
Little	34.20	19.18	16.98

Table 3.2: Estimated kinematic parameters of the hand model

Finger	c_x^w [mm]	c_y^w [mm]	c_z^w [mm]
Thumb	-11.84	13.96	32.24
Index	-7.902	-30.54	20.05
Medium	13.76	-27.27	21.60
Ring	34.32	-18.46	25.65
Little	52.37	-8.864	30.36

Table 3.3: Estimated position of each finger with respect to the palm-fixed frame w

In this work, the PCA is carried out on the data set consisting of the marker position, observed in a coordinate system fixed to the hand, during the calibration movement. The first two PCs identify the set of planes in which the markers move, while the third PC is the flexion axes, that is also the direction in which the variance of the data is minimum. In order to obtain a more noise-robust estimation, the PCA has been performed not only with one marker, but using all the markers placed on the finger.

After estimating the flexion axes, the direction of the adduction/abduction ones will be established according to the D-H convention.

Note that, for simplicity, the calibration procedure for only one finger will be described in the following. On the hand dorsum, three markers l, r, u, are displaced to define a reference system fixed to the hand and three markers, m_1, m_2, m_3 are displaced on each finger (see Fig. 3.3). Each marker m_i is fixed on the finger phalanx i.

In order to calibrate the angular sensors and the hand model, the following steps are performed (for a detailed description of the algorithm see Section 3.1.3):

1. for each time-frame, transformation of all the marker coordinates in a coordinate system $\boldsymbol{O}_w - \boldsymbol{x}_w\boldsymbol{y}_w\boldsymbol{z}_w$, fixed on the hand dorsum (see Fig. 3.3). In such a coordinate system, the marker m_1 moves along a circumference C_1

Finger	ϕ [deg]	ϑ [deg]	ψ [deg]
Thumb	51.20	52.61	-176.8
Index	9.099	5.442	-74.36
Medium	8.384	3.878	-71.77
Ring	10.82	3.833	-71.16
Little	9.462	2.733	-69.82

Table 3.4: Estimated orientation of the fingers with respect to the palm-fixed frame w expressed in RPY angles

2. use of the Principal Component Analysis (PCA) to reduce the 3D problem to a 2D problem (see Fig. 3.2). Through the PCA algorithm, a new coordinate system $\boldsymbol{O}_p - \boldsymbol{x}_p\boldsymbol{y}_p\boldsymbol{z}_p$ is defined, in which the marker trajectories not only lie in parallel planes, but have also the minimum variance along the $\boldsymbol{z}_p$ axis.

3. estimation, using a 2D LMS methods, of the center $\boldsymbol{c}_1$ of circumference C_1 expressed in the coordinate system p. $\boldsymbol{c}_1$ is supposed to be the position of the first finger joint

4. estimation of the second joint position $\boldsymbol{c}_2$. In the coordinate system fixed on the first link and having $\boldsymbol{c}_1$ as a origin, $\boldsymbol{c}_1 - \boldsymbol{x}_{2'}\boldsymbol{y}_{2'}\boldsymbol{z}_{2'}$, the marker $\boldsymbol{m}_2$ moves along a circumference C_2 with $\boldsymbol{c}_2$ as a center.

5. estimation, analogously, of the position and orientation of all the other hand joints. The position of the joint i is the center of rotation of the circumference i and the axis of rotation of the joint $i, i = 2, 3, 4$ is normal to the plane $\boldsymbol{x}_p\boldsymbol{y}_p$

6. computation of the DH parameters a_i as the distance of consecutive centers of rotation

7. computation of the joint angles and calibration curves of the angular sensors

8. computation of all the other parameters required by the algorithm for joint angle estimation (see Section 3.2 for more details)

Once the calibration curves of joint angular sensors have been constructed through the estimated angles $\vartheta_2(k)$, $\vartheta_3(k)$, $\vartheta_4(k)$, polynomial interpolators are adopted to obtain an analytic expression to be used in the sensor fusion algorithm (see Fig. 2.10, 2.11, 2.12). The procedure described above for computing $\vartheta_2(k)$, $\vartheta_3(k)$, $\vartheta_4(k)$ will be hereafter called RIK (Recursive Inverse Kinematics).

3.1.3 Details of the calibration algorithm

The section describes in details the algorithm for the hand model and joint angular sensors calibration.

The first step for hand and angular sensor calibration is to define the dorsum-fixed frame w. Such frame is chosen on the basis of the three markers fixed to the hand dorsum (the ochre markers in Fig. 1.2) whose pose with respect to the base frame b is represented by the matrix $\boldsymbol{T}_b^w(k)$, at the k-th frame, defined as

$$\boldsymbol{T}_w^b(k) = \begin{bmatrix} \boldsymbol{R}_w^b(k) & \boldsymbol{o}_w^b(k) \\ \mathbf{0^T} & 1 \end{bmatrix} \tag{3.1}$$

where

$$\begin{aligned} \boldsymbol{o}_w^b(k) &= \boldsymbol{r}_l^b(k) \\ \boldsymbol{R}_w^b(k) &= [\ \hat{\boldsymbol{x}}_w^b(k) \quad \hat{\boldsymbol{y}}_w^b(k) \quad \hat{\boldsymbol{z}}_w^b(k)\ ], \end{aligned}$$

being

$$\begin{aligned} \hat{\boldsymbol{x}}_w^b(k) &= \frac{\boldsymbol{r}_r^b(k) - \boldsymbol{r}_l^b(k)}{\|\boldsymbol{r}_r^b(k) - \boldsymbol{r}_l^b(k)\|} \\ \hat{\boldsymbol{z}}_w^b(k) &= \frac{(\ a(k) \quad b(k) \quad c(k)\ )^T}{\|(\ a(k) \quad b(k) \quad c(k)\ )\|}\ , \\ \hat{\boldsymbol{y}}_w^b(k) &= \hat{\boldsymbol{z}}_w^b(k) \times \hat{\boldsymbol{x}}_w^b(k) \end{aligned}$$

where $\hat{\boldsymbol{z}}_w$ is the unit vector normal to the plane defined by the ochre markers with coordinates $\boldsymbol{r}_l^b = \begin{bmatrix} r_{l_x} & r_{l_y} & r_{l_z} \end{bmatrix}^T$, $\boldsymbol{r}_r^b = \begin{bmatrix} r_{r_x} & r_{r_y} & r_{r_z} \end{bmatrix}^T$, $\boldsymbol{r}_u^b = \begin{bmatrix} r_{u_x} & r_{u_y} & r_{u_z} \end{bmatrix}^T$, and $a(k)$, $b(k)$, $c(k)$ are the components of a vector normal to this plane computed at the k-th frame as

$$a(k) = \det \begin{bmatrix} r_{r_y} - r_{l_y} & r_{r_z} - r_{l_z} \\ r_{u_y} - r_{l_y} & r_{u_z} - r_{l_z} \end{bmatrix} \tag{3.2}$$

$$b(k) = -\det \begin{bmatrix} r_{r_x} - r_{l_x} & r_{r_z} - r_{l_z} \\ r_{u_x} - r_{l_x} & r_{u_z} - r_{l_z} \end{bmatrix} \tag{3.3}$$

$$c(k) = \det \begin{bmatrix} r_{r_x} - r_{l_x} & r_{r_y} - r_{l_y} \\ r_{u_x} - r_{l_x} & r_{u_y} - r_{l_y} \end{bmatrix}. \tag{3.4}$$

The marker positions are expressed in the local coordinate system $\boldsymbol{O}_w - \boldsymbol{x}_w\boldsymbol{y}_w\boldsymbol{z}_w$ through the homogenous transformation (3.1) applied to the homogeneous coordinates of each marker, i.e.

$$\tilde{\boldsymbol{m}}_j^w(k) = \boldsymbol{T}_b^w(k)\tilde{\boldsymbol{m}}_j^b(k) \qquad j = 1, 2, 3, \tag{3.5}$$

where $k = 1, 2, ..., M$ is the time frame number and $\tilde{\boldsymbol{m}}_j^w = (m_{j_x}^w \quad m_{j_y}^w \quad m_{j_z}^w \quad 1)^T$ are the homogenous coordinates of the j-th marker referred to frame w defined in (3.1).

Since during the calibration movement all the markers of a finger move on parallel planes, using Principal Component Analysis [44] it is possible to define a reference frame $\boldsymbol{O}_p - \boldsymbol{x}_p\boldsymbol{y}_p\boldsymbol{z}_p$ in which the marker coordinates have significant variations only along $\boldsymbol{x}$ and $\boldsymbol{y}$, while along $\boldsymbol{z}$ are approximately constant. Let $\boldsymbol{R}_w^p$ be the rotation matrix from $\boldsymbol{O}_p - \boldsymbol{x}_p\boldsymbol{y}_p\boldsymbol{z}_p$ to $\boldsymbol{O}_w - \boldsymbol{x}_w\boldsymbol{y}_w\boldsymbol{z}_w$ and `pca` the algorithm that returns the matrix $\boldsymbol{R}_p^w = (\boldsymbol{R}_w^p)^T$, it is

$$\tilde{\boldsymbol{m}}_j^p = \boldsymbol{T}_w^p \tilde{\boldsymbol{m}}_j^w \qquad j = 1, 2, 3, \tag{3.6}$$

where

$$\begin{aligned} \boldsymbol{T}_w^p &= \begin{bmatrix} \boldsymbol{R}_p^{w^T} & \mathbf{0} \\ \mathbf{0^T} & 1 \end{bmatrix} \\ \boldsymbol{R}_p^w &= \texttt{pca}(\{\boldsymbol{m}_1^w(k) : \; k = 1, 2, ..., M\}). \end{aligned} \tag{3.7}$$

In the $\boldsymbol{O}_p - \boldsymbol{x}_p\boldsymbol{y}_p\boldsymbol{z}_p$ reference frame the marker m_1 moves along a circumference with centre $\boldsymbol{c}_1$ and radius r_1. Through a classical least square method it is possible to estimate $\boldsymbol{c}_1^p$ and r_1 (see, for example, [72], [61], [23]. Let `circle` be the algorithm which has in input the movement of a marker m^i in the reference system $\boldsymbol{O}_i - \boldsymbol{x}_i\boldsymbol{y}_i\boldsymbol{z}_i$ and in output the centre of rotation in the same coordinate system $\boldsymbol{c}^i$ and the radius r_i, then the center of rotation of the first joint is estimated as

$$(\boldsymbol{c}_1^p, r_1) = \texttt{circle}(\{\boldsymbol{m}_1^p(k) : \; k = 1, 2, ..., M\}). \tag{3.8}$$

Now, the second and third centers of rotation can be estimated according to the procedure below applied to all the time frames $k = 1, \ldots, M$. The procedure exploits the fact that both the frame $\boldsymbol{O}_i - \boldsymbol{x}_i\boldsymbol{y}_i\boldsymbol{z}_i$ and the frame $\boldsymbol{O}_{i'} - \boldsymbol{x}_{i'}\boldsymbol{y}_{i'}\boldsymbol{z}_{i'}$ are fixed to the link i, in particular the axis $\boldsymbol{x}_{i'}$ has the same direction of the segment joining the center of rotation $\boldsymbol{c}_i$ and the marker m_i

(see again Fig. 3.2).

$$\hat{\boldsymbol{x}}_{2'}^{p}(k) = \frac{\boldsymbol{m}_1^p(k) - \boldsymbol{c}_1^p}{\|\boldsymbol{m}_1^p(k) - \boldsymbol{c}_1^p\|} \tag{3.9}$$

$$\hat{\boldsymbol{z}}_{2'}^{p}(k) = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \tag{3.10}$$

$$\hat{\boldsymbol{y}}_{2'}^{p}(k) = \hat{\boldsymbol{z}}_{2'}^{p}(k) \times \hat{\boldsymbol{x}}_{2'}^{p}(k) \tag{3.11}$$

$$\boldsymbol{R}_{2'}^{p}(k) = \begin{pmatrix} \hat{\boldsymbol{x}}_{2'}^{p} & \hat{\boldsymbol{y}}_{2'}^{p} & \hat{\boldsymbol{z}}_{2'}^{p} \end{pmatrix} \tag{3.12}$$

$$\boldsymbol{T}_{p}^{2'}(k) = \begin{pmatrix} \boldsymbol{R}_{2'}^{p^T}(k) & -\boldsymbol{R}_{2'}^{p^T}(k)\boldsymbol{c}_1^p \\ \mathbf{0^T} & 1 \end{pmatrix} \tag{3.13}$$

$$\tilde{\boldsymbol{m}}_{j}^{2'}(k) = \boldsymbol{T}_{p}^{2'}\boldsymbol{m}_{j}^{p}(k) \qquad j = 1,2,3 \tag{3.14}$$

$$(\boldsymbol{c}_2^{2'}, r_2) = \texttt{circle}(\{\boldsymbol{m}_2^{2'} : k = 1,2,...,M\}) \tag{3.15}$$

$$\bar{\vartheta}_2 = \text{atan2}(\boldsymbol{e}_2^T\boldsymbol{c}_2^{2'}, \boldsymbol{e}_1^T\boldsymbol{c}_2^{2'}) \tag{3.16}$$

$$\boldsymbol{T}_{2'}^{2} = \begin{pmatrix} \boldsymbol{R}_z(\bar{\vartheta}_2) & -\boldsymbol{R}_z(\bar{\vartheta}_2)c_2^{2'} \\ \mathbf{0^T} & 1 \end{pmatrix} \tag{3.17}$$

$$\tilde{\boldsymbol{m}}_{j}^{2} = \boldsymbol{T}_{2'}^{2}\tilde{\boldsymbol{m}}_{j}^{2'} \qquad j = 2,3 \tag{3.18}$$

$$\hat{\boldsymbol{x}}_{3'}^{2}(k) = \frac{\boldsymbol{m}_2^2(k)}{\|\boldsymbol{m}_2^2(k)\|} \tag{3.19}$$

$$\hat{\boldsymbol{z}}_{3'}^{2}(k) = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \tag{3.20}$$

$$\hat{\boldsymbol{y}}_{3'}^{2}(k) = \hat{\boldsymbol{z}}_{3'}^{2}(k) \times \hat{\boldsymbol{x}}_{3'}^{2}(k) \tag{3.21}$$

$$\boldsymbol{R}_{3'}^{2}(k) = \begin{pmatrix} \hat{\boldsymbol{x}}_{3'}^{2} & \hat{\boldsymbol{y}}_{3'}^{2} & \hat{\boldsymbol{z}}_{3'}^{2} \end{pmatrix} \tag{3.22}$$

$$\boldsymbol{T}_{2}^{3'}(k) = \begin{pmatrix} \boldsymbol{R}_{3'}^{2^T}(k) & 0 \\ \mathbf{0^T} & 1 \end{pmatrix} \tag{3.23}$$

$$\tilde{m}_{3}^{3'}(k) = \boldsymbol{T}_{2}^{3'}(k)\tilde{m}_{3}^{2}(k) \tag{3.24}$$

$$(\boldsymbol{c}_3^{3'}, r_3) = \texttt{circle}(\{\boldsymbol{m}_3^{3'} : k = 1,2,...,M\}) \tag{3.25}$$

$$\bar{\vartheta}_3 = \text{atan2}(\boldsymbol{e}_2^T\boldsymbol{c}_3^{3'}, \boldsymbol{e}_1^T\boldsymbol{c}_3^{3'}) \tag{3.26}$$

$$\boldsymbol{T}_{3'}^{3} = \begin{pmatrix} \boldsymbol{R}_z(\bar{\vartheta}_3) & -\boldsymbol{R}_z(\bar{\vartheta}_3)\boldsymbol{c}_3^{3'} \\ \mathbf{0^T} & 1 \end{pmatrix} \tag{3.27}$$

The relations (3.9)–(3.14) are used to express the pose of the frame $2'$ with respect to the frame p and to refer the coordinates of the markers to the frame $2'$. Equation (3.15) estimates the second center of rotation. Relations (3.16)–(3.17) are aimed at aligning frame $2'$ with frame 2. Equation (3.18) is needed to express markers m_2 and m_3 with respect to frame 2, while relations (3.19)–(3.23) estimate the pose of frame $3'$. Equation (3.24) expresses marker m_3 in frame $3'$, while equation (3.25) estimates the third center of rotation. The last two equations (3.26)–(3.27) align frame $3'$ with frame 3.

Now all the kinematic model parameters can be computed, namely the link lengths and the coordinates of the markers in the plane of the finger structure, as

$$a_2 = \|\boldsymbol{c}_2^{2'}\| \tag{3.28}$$

$$a_3 = \|\boldsymbol{c}_3^{3'}\| \tag{3.29}$$

$$a_4 = r_3 \tag{3.30}$$

$$\begin{pmatrix} l_1 & h_1 \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{e}_1^T \boldsymbol{m}_1^2 & \boldsymbol{e}_2^T \boldsymbol{m}_1^2 \end{pmatrix}^T \tag{3.31}$$

$$\begin{pmatrix} l_2 & h_2 \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{e}_1^T \boldsymbol{m}_2^3 & \boldsymbol{e}_2^T \boldsymbol{m}_2^3 \end{pmatrix}^T \tag{3.32}$$

Note that, with respect to any frame with the $\boldsymbol{z}$ axis aligned with $\boldsymbol{z}_p$, the z coordinates are assumed zero since for the determination of joint angles corresponding to flexion movements only x and y coordinates are needed and during the calibration session it is supposed $\vartheta_1(k) = 0,\ \forall k$. This implies that the calibration hand movement has to be performed without moving the abduction DOF. The estimated x and y coordinates of the markers are denoted with (l_i, h_i).

Note that, to avoid the necessity to perform also abduction movements which are characterized by small ranges of motion, it is assumed that, when the finger is completely extended, the angle ϑ_2 is 0, therefore the abduction axis is orthogonal both to the axis $\boldsymbol{z}_1$ of joint 2 and to the axis $\boldsymbol{x}_2$, which is aligned with $\boldsymbol{x}_1$ in such a configuration (see Fig. 3.2). This means that the relative position and orientation of frame 1 with respect to frame p is given by

$$\boldsymbol{T}_1^p(k) = \boldsymbol{T}_1^p(1) = \begin{bmatrix} \boldsymbol{R}_{2'}^p(1)\boldsymbol{R}_2^{2'} & \boldsymbol{c}_1^p \\ \mathbf{0^T} & 1 \end{bmatrix},\ \forall\, k \tag{3.33}$$

where the assumption that ϑ_1 is always 0 has been taken into account.

3.2 Fusion algorithm design

Since the motion capture system presents the problem of marker occlusion and the angular sensors are less accurate and can not be applied to all the hand DOFs, the sensor fusion seems to be the best approach to tackle the problem of the real-time observation of human manipulation. The first step to deal with a Bayesian sensor fusion problem is defining a stochastic model of the system. Then, with the aid of computer simulations, an EKF-based sensor fusion algorithm has been designed.

3.2.1 System modeling

To limit the notation complexity, the algorithm will be presented for a single finger. Firstly, define a state-space model of the system with state vector

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\vartheta} & \dot{\boldsymbol{\vartheta}} & \boldsymbol{l} & \boldsymbol{h} \end{pmatrix}^T \tag{3.34}$$

being

$$\begin{aligned} \boldsymbol{\vartheta} &= \begin{pmatrix} \vartheta_1 & \vartheta_2 & \vartheta_3 & \vartheta_4 \end{pmatrix}^T && (3.35) \\ \dot{\boldsymbol{\vartheta}} &= \begin{pmatrix} \dot{\vartheta}_1 & \dot{\vartheta}_2 & \dot{\vartheta}_3 & \dot{\vartheta}_4 \end{pmatrix}^T && (3.36) \\ \boldsymbol{l} &= \begin{pmatrix} l_1 & l_2 \end{pmatrix}^T && (3.37) \\ \boldsymbol{h} &= \begin{pmatrix} h_1 & h_2 \end{pmatrix}^T && (3.38) \end{aligned}$$

where (l_1, h_1) and (l_2, h_2) are the (x, y) coordinates of markers m_1 and m_2 in the frames $\boldsymbol{c}_2 - \boldsymbol{x}_2\boldsymbol{y}_2\boldsymbol{z}_2$ and $\boldsymbol{c}_3 - \boldsymbol{x}_3\boldsymbol{y}_3\boldsymbol{z}_3$ respectively and have been considered as state variables to be estimated, instead of system parameters, to take into account the sliding of the markers with respect to the finger bones.

As usual in tracking problems tackled via state estimation techniques, in the state update model, the joint variables to be tracked are assumed to vary with a constant velocity, while the unknown parameters are assumed to be constant. Hence the state update equations are

$$\begin{aligned} \boldsymbol{\vartheta}_{k+1} &= \boldsymbol{\vartheta}_k + \dot{\boldsymbol{\vartheta}}_k \Delta t + \boldsymbol{w}_k^{\vartheta} \\ \dot{\boldsymbol{\vartheta}}_{k+1} &= \dot{\boldsymbol{\vartheta}}_k + \boldsymbol{w}_k^{\dot{\vartheta}} \\ \boldsymbol{l}_{k+1} &= \boldsymbol{l}_k + \boldsymbol{w}_k^{l} \\ \boldsymbol{h}_{k+1} &= \boldsymbol{h}_k + \boldsymbol{w}_k^{h} \end{aligned} \tag{3.39}$$

with Δt the sampling time of the filter, chosen as the minimum of the sampling times of the sensors, in case the sensors have different sampling rates.

It is easy to see that Eq. (3.39) is linear, then it can be written in matrix form as

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}\boldsymbol{x}_k + \boldsymbol{w}_k \tag{3.40}$$

where

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_4 & \Delta t \boldsymbol{I}_4 & \boldsymbol{0_4} \\ \boldsymbol{0_4} & \boldsymbol{I}_4 & \boldsymbol{0_4} \\ \boldsymbol{0_4} & \boldsymbol{0_4} & \boldsymbol{I}_4 \end{bmatrix}, \tag{3.41}$$

and

$$\boldsymbol{w}_k = \begin{bmatrix} (\boldsymbol{w}_k^{\vartheta})^T & (\boldsymbol{w}_k^{\dot{\vartheta}})^T & (\boldsymbol{w}_k^{l})^T & (\boldsymbol{w}_k^{h})^T \end{bmatrix}^T \tag{3.42}$$

is a 12×1 vector of stochastic processes. The hypothesis under the Kalman-like filters is that n_k and w_k are additive Gaussian white noise (AGWN). The measurement equation is

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{m}_1^b(\boldsymbol{x}_k) \\ \boldsymbol{m}_2^b(\boldsymbol{x}_k) \\ \boldsymbol{m}_3^b(\boldsymbol{x}_k) \\ v_2(\boldsymbol{x}_k) \\ v_3(\boldsymbol{x}_k) \\ v_4(\boldsymbol{x}_k) \end{bmatrix} + \boldsymbol{n}_k \triangleq \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{n}_k \tag{3.43}$$

where $v_i(x)$ is the voltage measured by the angular sensor applied to i-th joint and its analytic expression has been determined in Section 3.1.2. The marker positions with respect to the D-H frame 0 are computed as

$$\tilde{\boldsymbol{m}}_1^b = \boldsymbol{T}_w^b(k)\boldsymbol{T}_p^w\boldsymbol{T}_0^p\boldsymbol{T}_2^0(k)\tilde{\boldsymbol{m}}_1^2 \tag{3.44}$$

$$\tilde{\boldsymbol{m}}_2^b = \boldsymbol{T}_w^b(k)\boldsymbol{T}_p^w\boldsymbol{T}_0^p\boldsymbol{T}_3^0(k)\tilde{\boldsymbol{m}}_2^3 \tag{3.45}$$

$$\tilde{\boldsymbol{m}}_3^b = \boldsymbol{T}_w^b(k)\boldsymbol{T}_p^w\boldsymbol{T}_0^p\boldsymbol{T}_4^0(k)\tilde{\boldsymbol{m}}_3^4 \tag{3.46}$$

where $\boldsymbol{T}_w^b(k)$ is computed as in Eq. (3.1), $\boldsymbol{T}_p^w$ as in Eq. (3.7), $\boldsymbol{T}_i^{i-1}(k), \quad i = 0, 1, 2, 3, 4$ are the D-H transformations, $\boldsymbol{T}_0^p$ can be computed as

$$\boldsymbol{T}_o^p = \boldsymbol{T}_1^p(1) \begin{bmatrix} \boldsymbol{R}_x(\pi/2) & 0 \\ \mathbf{0^T} & 1 \end{bmatrix},$$

being $\boldsymbol{T}_1^p(1)$ the constant matrix calculated in Eq. (3.33) during the calibration phase, and, by definition,

$$\begin{aligned} \tilde{\boldsymbol{m}}_1^2 &= \begin{pmatrix} l_1 & h_1 & 0 & 1 \end{pmatrix} \\ \tilde{\boldsymbol{m}}_2^3 &= \begin{pmatrix} l_2 & h_2 & 0 & 1 \end{pmatrix} \\ \tilde{\boldsymbol{m}}_3^4 &= \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

According to the Kalman filter framework, state variables are assumed to be Gaussian stochastic processes, with zero mean and diagonal covariance matrix, set by aid of computer simulations. The hypothesis of Gaussian pdf measurement error, on which the Kalman-like filters are based, is a restrictive hypothesis, since the measurement model is strongly nonlinear and the angular sensor characteristics may be not accurate. An improvement of filter performance can be obtained through more complex filtering techniques like particle filters [63], which allow to estimate effectively the state of nonlinear systems even if the noise pdf is not Gaussian. The aim of the Extended Kalman Filter is to track joint angular positions and velocities

in a way robust to occlusions and marker slipping phenomena. Occlusion marker problem is handled by fusion of camera measurements and angular sensors measurements; the sensor fusion algorithms improve the measurement system robustness, since one sensor can contribute information while others are unavailable, jammed, or lack coverage of a target or event. The marker slipping phenomenon is handled by inserting in the state vector the marker positions expressed in the reference frames fixed to the finger links, hence they are estimated by the Extended Kalman Filter as well. To model a system for filtering or data fusion purposes it is very important to evaluate the variance of measurement errors. If the error variance of the sensors is badly evaluated, the sensor fusion algorithm may become ineffective. It is assumed that the measurement noises are independent from each other, hence the covariance matrix is assumed diagonal. The experimentally estimated variances of camera measurement noises are

$$\begin{aligned} \sigma_{m_1}^2 &= 9 \cdot 10^{-6}\ \mathrm{m}^2 \\ \sigma_{m_2}^2 &= 9 \cdot 10^{-6}\ \mathrm{m}^2 \\ \sigma_{m_3}^2 &= 9 \cdot 10^{-6}\ \mathrm{m}^2 \end{aligned} \tag{3.47}$$

assumed equal in all the directions. Whereas, the experimentally estimated variances of angular sensor noises are

$$\begin{aligned} \sigma_{v_2}^2 &= 3.5 \cdot 10^{-5}\ \mathrm{V}^2 \\ \sigma_{v_3}^2 &= 4.7 \cdot 10^{-5}\ \mathrm{V}^2 \\ \sigma_{v_4}^2 &= 2.3 \cdot 10^{-5}\ \mathrm{V}^2 \end{aligned} \tag{3.48}$$

3.2.2 Filter design

Whenever a marker is occluded, the model of the system changes, then also the sensor fusion algorithm parameters have to change in order to estimate effectively the joint angles. The whole system is then modelled as a switching nonlinear system, in which each state of a finite state machine matches a nonlinear set of differential equations describing the system. The state of the finite state machine is represented by the variable S_{ijk}, where i, j, k are equal to zero if the marker m_1, m_2, m_3, respectively, is not occluded and they are equal to one otherwise. As a consequence, to cope with the problem of marker occlusion, a switching EKF (SEKF) has been proposed. When no marker occlusion occurs, i.e. when the system is in the state S_{000}, we can design the Extended Kalman Filter by setting the covariance measurement matrix $\boldsymbol{Q}$ as a diagonal 12×12 matrix, having measurement noise covariance as diagonal element

$$\boldsymbol{Q} = \mathrm{diag}\{\sigma_{m_1}^2 \boldsymbol{I}_3, \sigma_{m_2}^2 \boldsymbol{I}_3, \sigma_{m_3}^2 \boldsymbol{I}_3, \sigma_{v_2}^2, \sigma_{v_3}^2, \sigma_{v_4}^2\}. \tag{3.49}$$

Instead, it is more difficult to choose the model covariance matrix $\boldsymbol{V}$; the simplest method, adopted here, is to set the model covariance matrix as a diagonal matrix, i.e. to assume that the process modelling errors are mutually independent. In [63] a set of empirical and semi-empirical methods for selecting the V matrix entries are described, , while [56] proposes a method which ensures convergence of the filter. The entries have been set through the aid of computer simulations; the rule of thumb most frequently adopted for tuning the $\boldsymbol{V}$ matrix is that decreasing the value of its elements, implies that the filter "bandwidth" decreases and the measurement noise is attenuated; increasing the values of $\boldsymbol{V}$ entries, process modelling error is attenuated and the filter "bandwidth" increases. The performed simulations show that the filter presents good performance when $\boldsymbol{V}$ is set as

$$\begin{aligned}\boldsymbol{V} \;=\; \mathrm{diag}\{&5\cdot 10^{-8}, 5\cdot 10^{-8}, 1\cdot 10^{-7}, 1\cdot 10^{-6}, 5\cdot 10^{-8}, 5\cdot 10^{-8},\\ &10^{-6}, 10^{-7}, 5\cdot 10^{-11}, 10^{-11}, 10^{-11}, 10^{-11}\}\end{aligned}$$

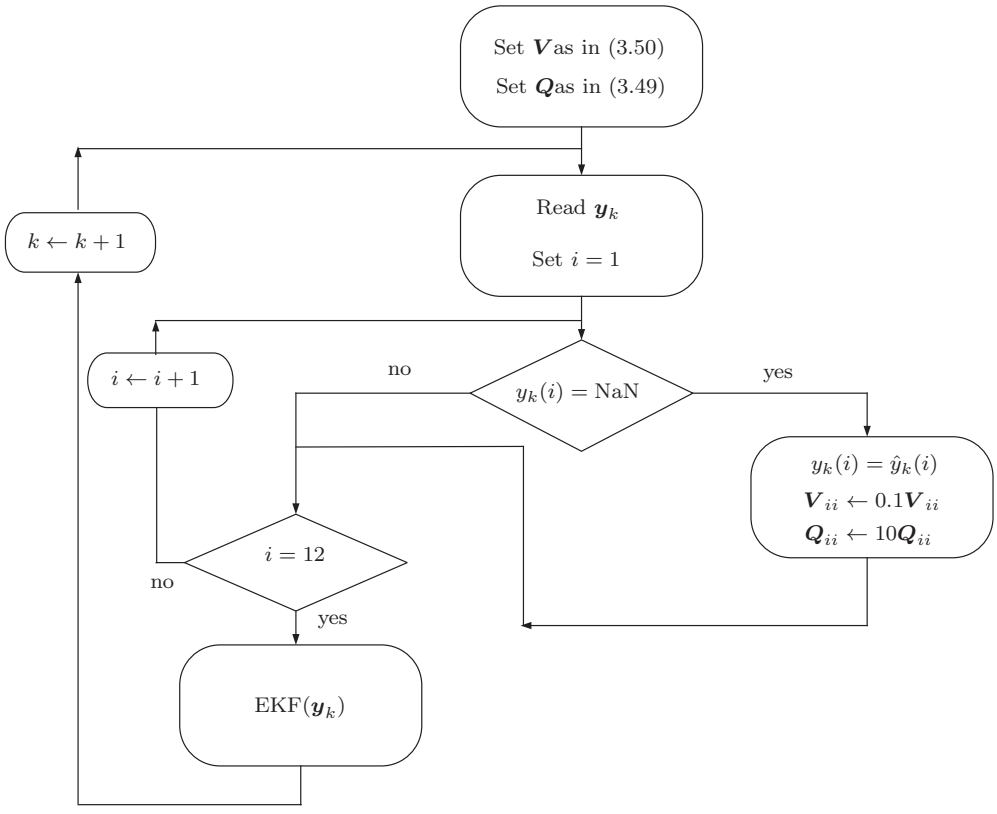


Figure 3.4: Flowchart of the switching EKF

In presence of marker occlusions (i.e. when the finite state machine is not in state S_{000}), the filter receives in input the value `NaN`. Computer simulation has been used also to tune the filter in presence of occlusions, by using a circular movement in 2D space. It is important to specify that in the general case of a movement in 3D space, since no abduction angular sensor is present, measurement of ϑ_1 is not possible and thus accurate tracking of the finger abduction movement is not possible in presence of permanent occlusion. The algorithm to switch from an EKF model to another is based on camera measurement observations. In detail, when $\boldsymbol{y}_k^{m_i} = \texttt{NaN}$, the filter sets

$$\boldsymbol{y}_k^{m_i} = \hat{\boldsymbol{y}}_k^{m_i},$$

where $\boldsymbol{y}_k^{m_i}$ is the vector containing the components of $\boldsymbol{y}_k$ corresponding to marker m_i and $\hat{\boldsymbol{y}}_k^{m_i}$ is its estimate computed by taking the same components of the vector $\boldsymbol{h}(\hat{\boldsymbol{x}}_{k-1})$ in (3.43), with $\hat{\boldsymbol{x}}_{k-1}$ the estimated state at step $k-1$. Moreover, since the angular sensor measurements are less reliable than the camera measurements, the model covariance matrix is modified as

$$\boldsymbol{V}_{\text{occlusion}}^{m_i} = 0.1 \cdot \boldsymbol{V}^{m_i}, \tag{3.50}$$

$$\boldsymbol{Q}_{\text{occlusion}}^{m_i} = 10 \cdot \boldsymbol{Q}^{m_i}, \tag{3.51}$$

where $\boldsymbol{V}_{\text{occlusion}}^{m_i}$ and $\boldsymbol{Q}_{\text{occlusion}}^{m_i}$ are the sub-matrices of $\boldsymbol{V}_{\text{occlusion}}$ and $\boldsymbol{Q}_{\text{occlusion}}$ containing the elements corresponding to the occluded marker m_i, being $\boldsymbol{V}_{\text{occlusion}}$ and $\boldsymbol{Q}_{\text{occlusion}}$ the model and the measurement covariance matrices, respectively, when an occlusion happens. Whereas, $\boldsymbol{V}^{m_i}$ and $\boldsymbol{Q}^{m_i}$ are the sub-matrices of $\boldsymbol{V}$ and $\boldsymbol{Q}$ which contain the elements corresponding to the marker m_i when the marker is not occluded. The algorithm flow chart is sketched in Fig. 3.4.

Simulation Results

To test the filter, the Simulink model in Fig. 3.5 was used. The main subsystems of the model are

- Motion Planner: it generates a planar circular finger movement in the operational space, in this case the trajectory of the marker m_3 placed on the finger tip.
- Inverse Kinematics: it computes the angular joint variable trajectories ϑ_2, ϑ_3, ϑ_4 from the positions in the Cartesian space of m_3. The trajectory ϑ_1 is generated by the block `thetar`.
- Block `thetar`: it generates the trajectory of abduction joint ϑ_1

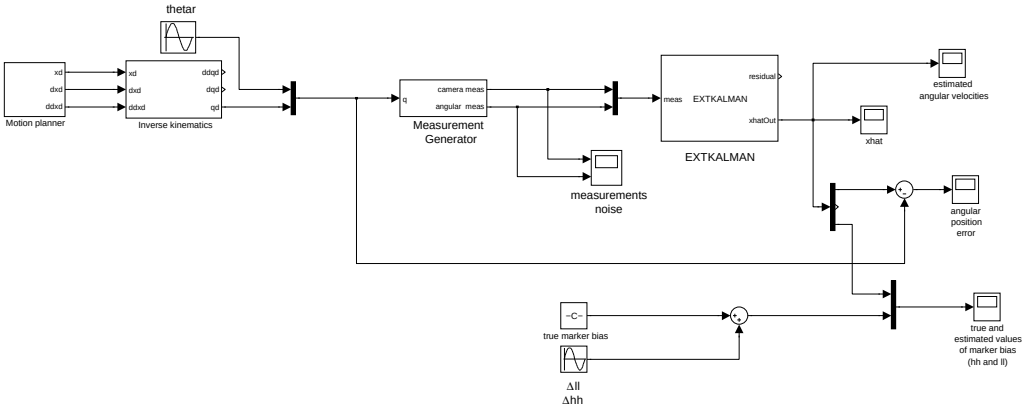


Figure 3.5: Simulink block scheme for evaluating the performance of EKF

- Measurement Generator: it generates, from the vector $\begin{pmatrix}\vartheta_1 & \vartheta_2 & \vartheta_3 & \vartheta_4\end{pmatrix}$, the simulated noisy measurements of cameras and of angular sensors, at frequencies of 60 Hz and 240 Hz respectively.

- `EXTKALMAN`: it implements the switching EKF designed, which, from the voltage values of angular sensors and marker positions from the cameras, computes the estimated joint angular positions. The `EXTKALMAN` block has been implemented by using the Embedded Matlab Toolbox[1].

Simulations have been performed in the following three case studies, setting the filter initial conditions to a random value

1. **Observation without marker occlusion**. Figure 3.6, showing true and estimated joint angles, demonstrates that the filter correctly estimates the variables in presence of noise. Figure 3.7 reports also the estimated joint velocities even without any direct velocity measurement. Figure 3.8 reports the estimated marker positions with respect to the frames fixed to the corresponding links. The filter usefully exploits the available measurements (see Fig. 3.13) to estimate quite satisfactorily the slipping motion.

2. **Observation with periodic occlusion of markers m_1 and m_3**. Figure 3.9 shows the true and estimated joint angles when a periodic occlusion lasting for 250 ms affects both the markers m_1 and m_3. In spite of the long lasting occlusions, the filter is still able to reconstruct

[1]Embedded MATLAB is a subset of the MATLAB language that allows to generate production quality C code for embedded applications; it restricts MATLAB semantics to meet the memory and data type requirements of embedded target environments.

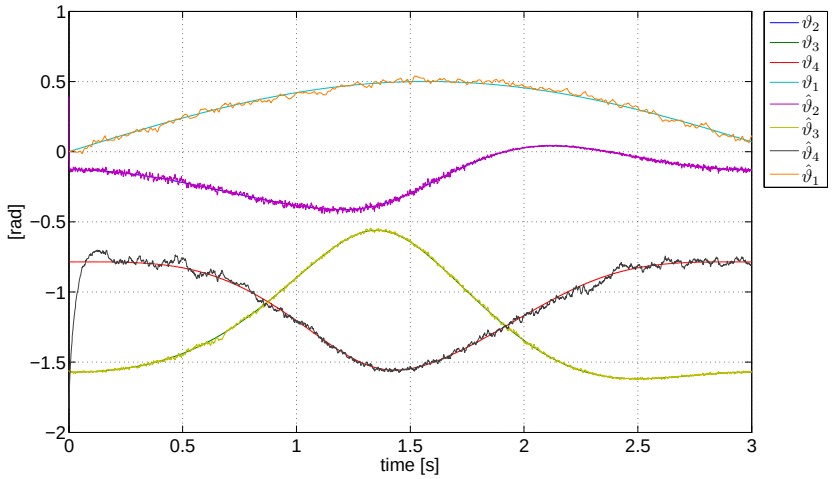


Figure 3.6: True and estimated joint angles (without marker occlusion)

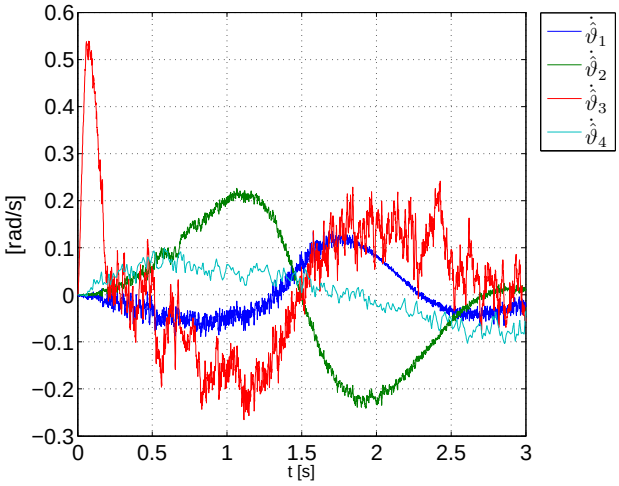


Figure 3.7: Estimated joint velocities (without marker occlusion)

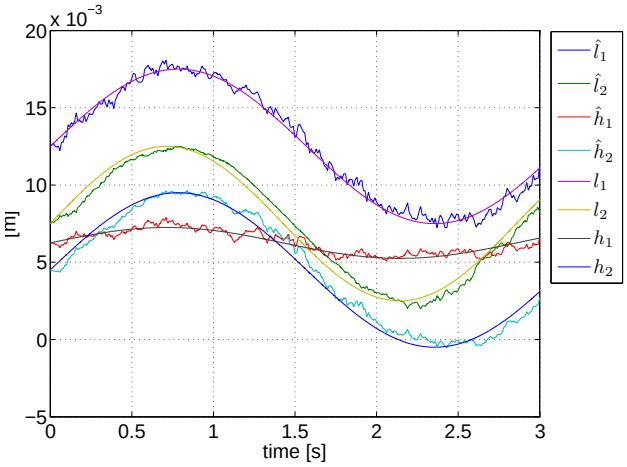


Figure 3.8: Marker slipping without marker occlusion

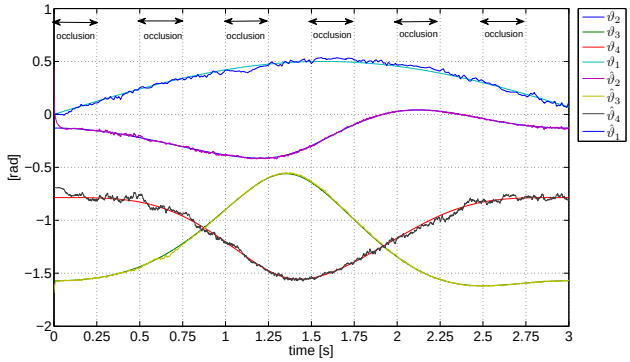


Figure 3.9: True and estimated joint angles with m_1 and m_3 periodically occluded

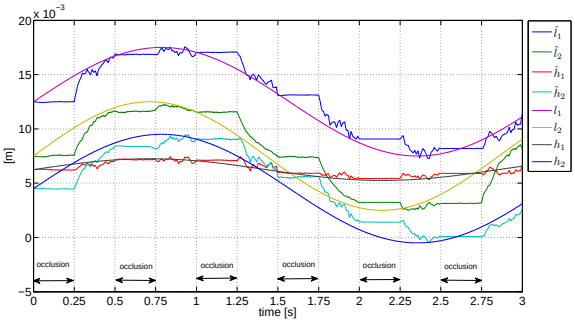


Figure 3.10: True and estimated marker slipping with m_1 and m_3 periodic occlusion

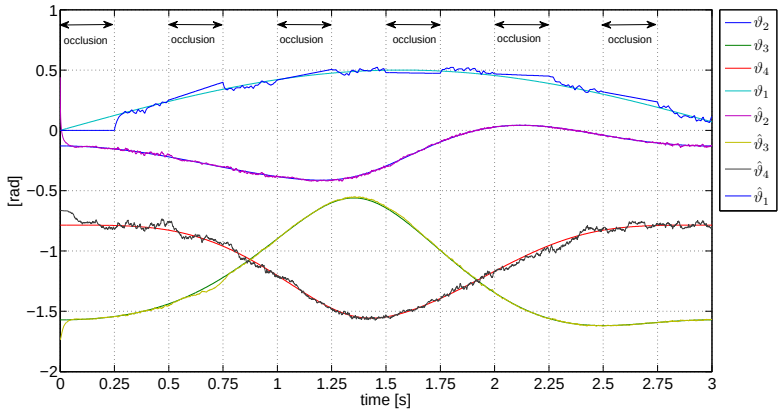


Figure 3.11: True and estimated angular joint positions in presence of periodical simultaneous occlusions of all markers

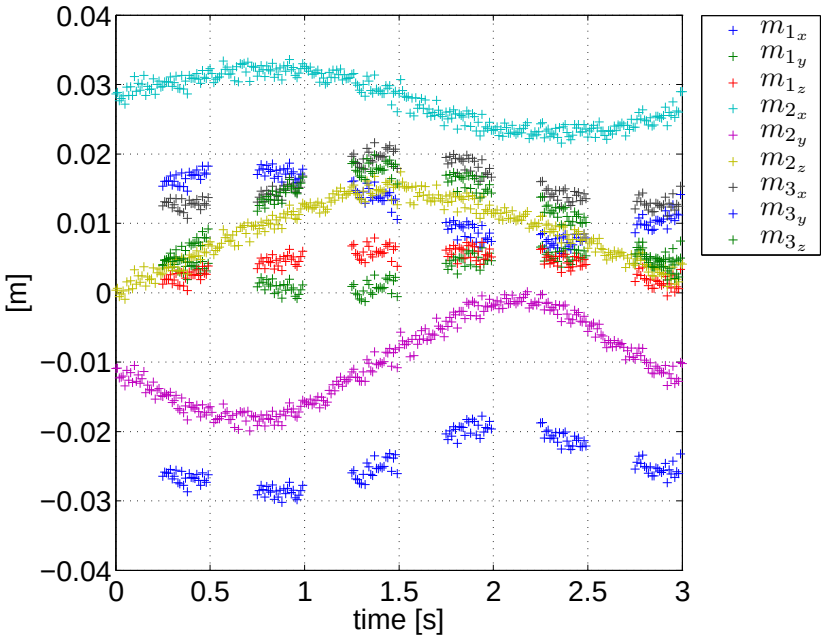


Figure 3.12: Marker positions measured by cameras with periodic occlusion of the markers m_1 and m_3.

quite accurately the joint angles. Figure 3.10 reports the estimated marker slipping motion and it is evident how the performance of the filter is worse than in the previous case study since the information on the marker position resides only in the camera measurements, which are reported in Fig. 3.12. Nevertheless, owing to the dynamic nature of the filter, the estimated position, kept constant during occlusions, tracks the actual position as soon as the occlusion terminates.

3. **Observation with periodic occlusion of all the markers**. This case study intends to show that when the motion capture system provides no information, only a subset of the joint angles can be correctly estimated. In fact, Fig. 3.11 shows that during occlusions, the abduction joint angle (ϑ_1) is always kept constant since no angle sensor is mounted for that joint. Whereas, the other joint angles are accurately estimated.

Figure 3.12 presents the noisy camera measurements generated in simulation from the true trajectory and evidences the occlusion phenomenon and the different sampling rates of camera and optical sensors. The measurement noise power is set one order of magnitude greater than nominal motion capture system noise, estimated in [64]. Figure 3.13 presents the noisy angular sensor measurements generated in simulation from the true trajectory of $\boldsymbol{\vartheta}$. The measurement noise power used in simulation is one order of magnitude greater than the nominal angular sensors noise, estimated in [64].

3.3 Results and discussion

The switching EKF has been tested experimentally. A preliminary set of experiments consists of a finger extension and flexion movements on a plane. During the movements marker occlusion phenomenon occurs.

The joint angle trajectories, obtained by the EKF, are compared with the trajectories obtained by the RIK algorithm developed to calibrate the angular sensors and described in Section 3.1.2. The algorithm off-line estimates the angular position variables from the position of the marker measured by the motion capture system and it is not robust to occlusion, since it does not use the angular sensors. A comparison with the main commercial solutions for joint angle estimation was not possible, since they require three markers per link and an impractical number of cameras would be needed to perform such experiments.

Figures 3.14, 3.15, 3.16, 3.17 show the angular position of the joint 1, 2, 3 and 4, respectively, estimated by the EKF sensor fusion algorithm (blue

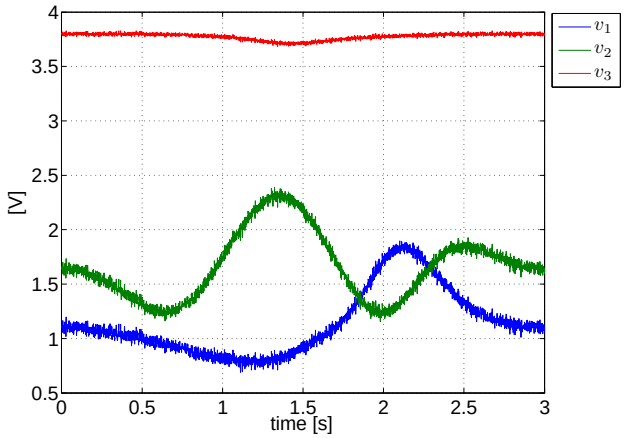


Figure 3.13: Angular sensor measurements (equals in each simulation)

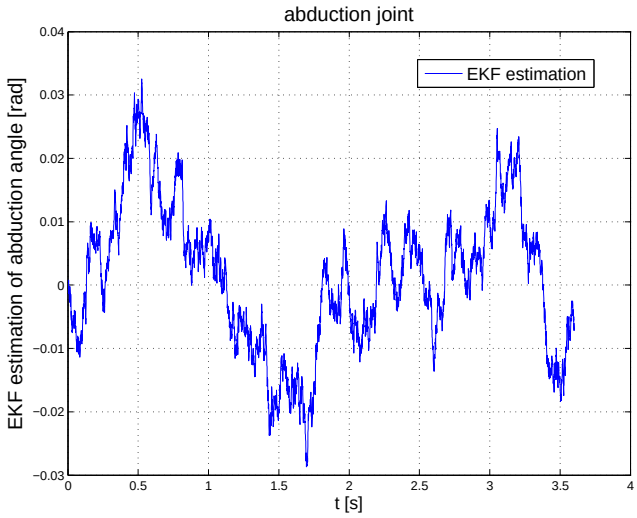


Figure 3.14: Abduction joint angle position (ϑ_1)

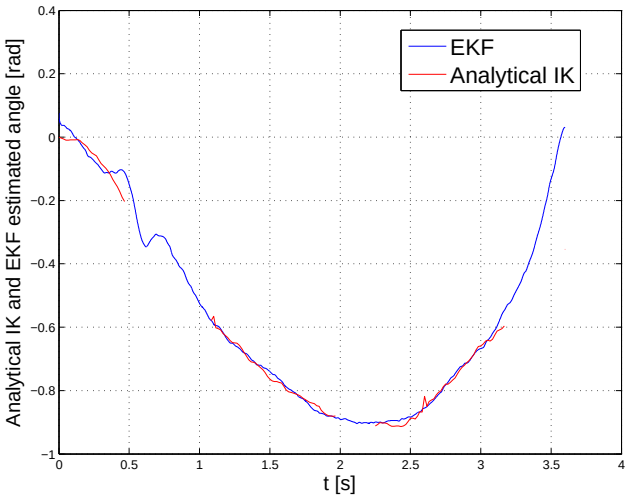


Figure 3.15: Flexion joint angle position (ϑ_2) estimated through proposed EKF (blue line) and through analytical inversion kinematic algorithm (red curve).

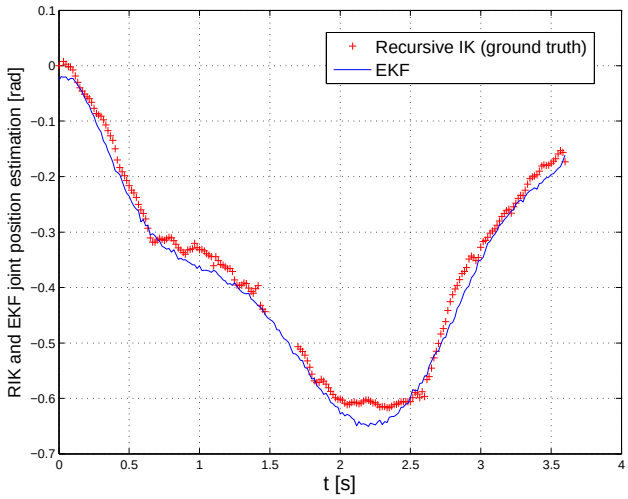


Figure 3.16: Joint 3 angle position (ϑ_3) estimated through proposed EKF (blue line) and through analytical inversion kinematic algorithm (red curve).

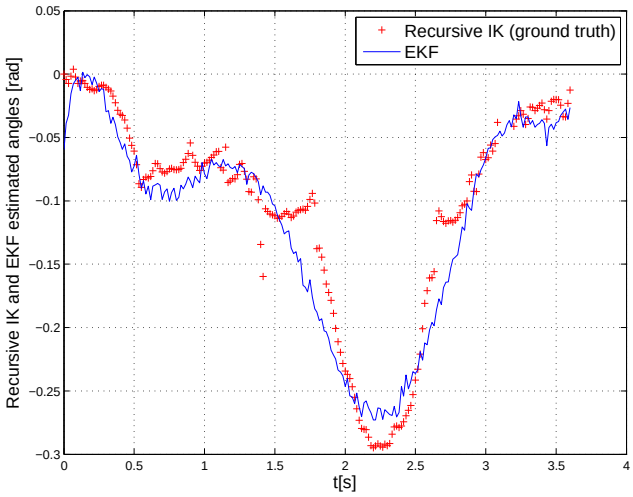


Figure 3.17: Joint 4 angle position (ϑ_4) estimated through proposed EKF (blue line) and through recursive inversion kinematic algorithm (red line).

line) and by the RIK Inversion algorithm (red line).

Fig. 3.14 shows the EKF estimation of the abduction angle ϑ_1. The RIK algorithm does not estimate ϑ_1, because it uses a simplified model of the human hand and it can consider only finger movements without the abduction DoF.

In presence of occlusions, the EKF allows obtaining a real-time smooth fill gap.

3.3.1 Repository of captured data

After the encouraging results of the preliminary experiments, a repository of observed grasp and manipulation tasks has been set up. For each trial, it contains estimations of the hand pose in the space, joint angles and normal components of fingertip contact forces.

The grasp and manipulation tasks have been performed on a set of elementary objects, which includes a cup, a pencil, a soft empty bottle, a soft half-filled bottle, a small empty rigid bottle and a large empty rigid bottle. Some objects can be distinguished only through contact force measurements, e.g. soft empty bottle and soft half-filled bottle. This choice has been made in order to empathize the importance of contact force knowledge in the observation of human grasping and manipulation.

The gathered data has been used within Dexmart [2] to tackle several

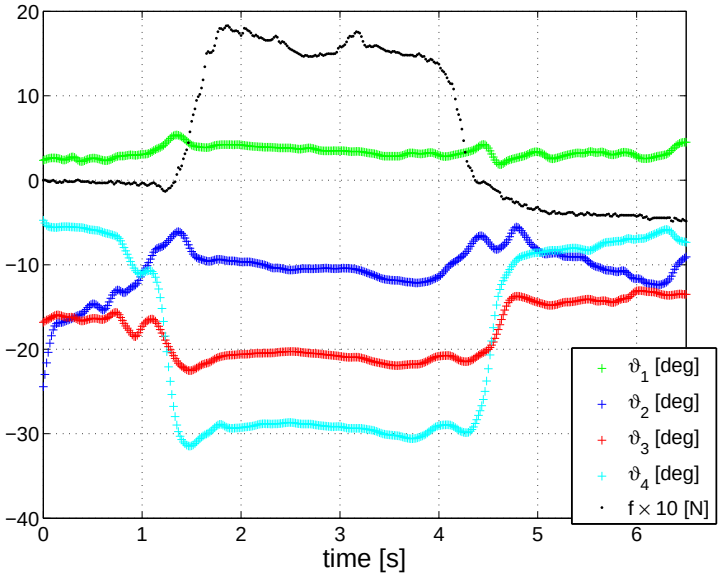


Figure 3.18: Index joint angles and estimated tactile forces for the parallelepiped shown in Fig. 3.26 (a)

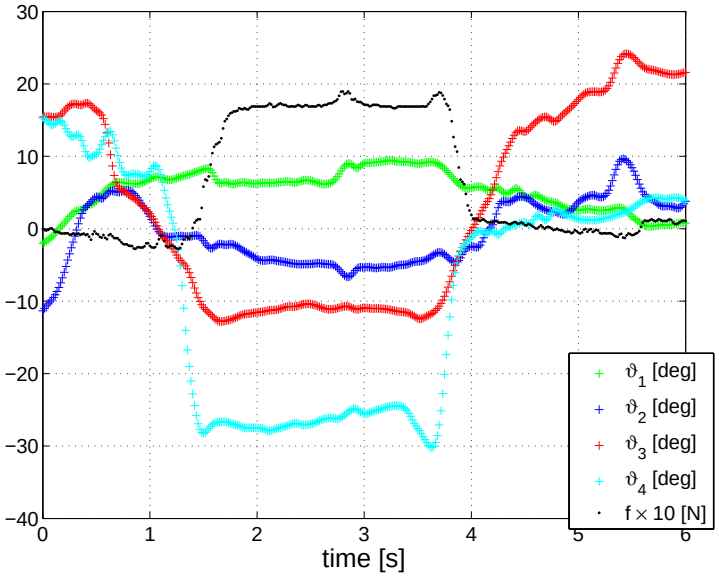


Figure 3.19: Index joint angles and estimated tactile forces for the empty plastic bottle in Fig. 3.26 (b)

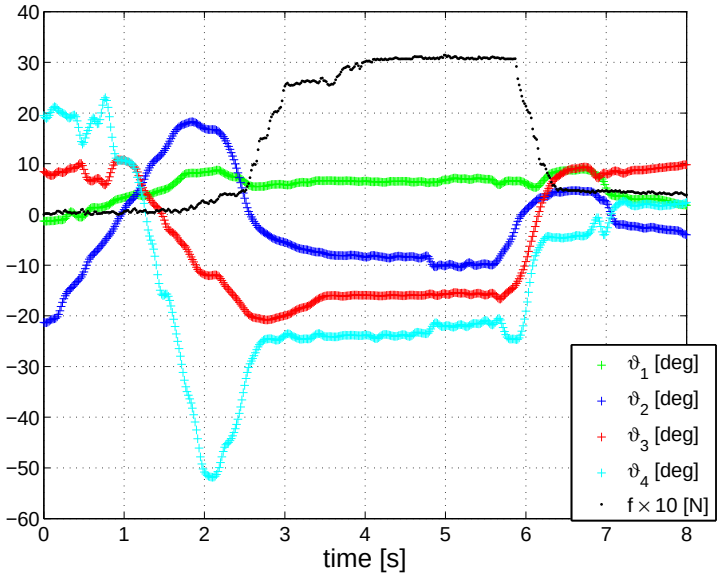


Figure 3.20: Index joint angles and estimated tactile forces for the half-filled plastic bottle shown in Fig. 3.26 (b)

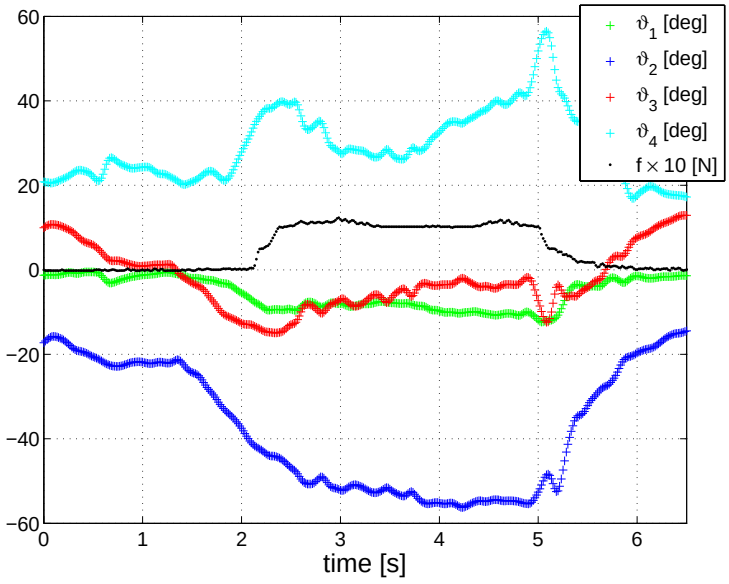


Figure 3.21: Index joint angles and estimated tactile forces for the bottle in Fig. 3.27 (a)

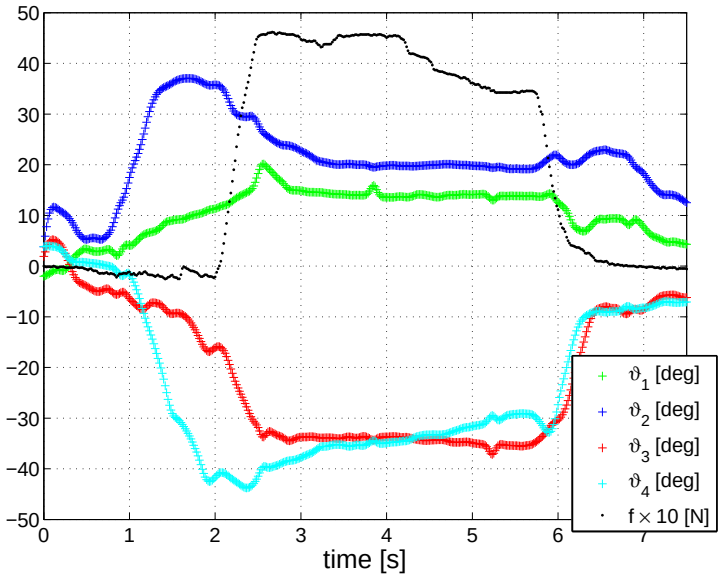


Figure 3.22: Index joint angles and estimated tactile forces for the bottle in Fig. 3.27 (b)

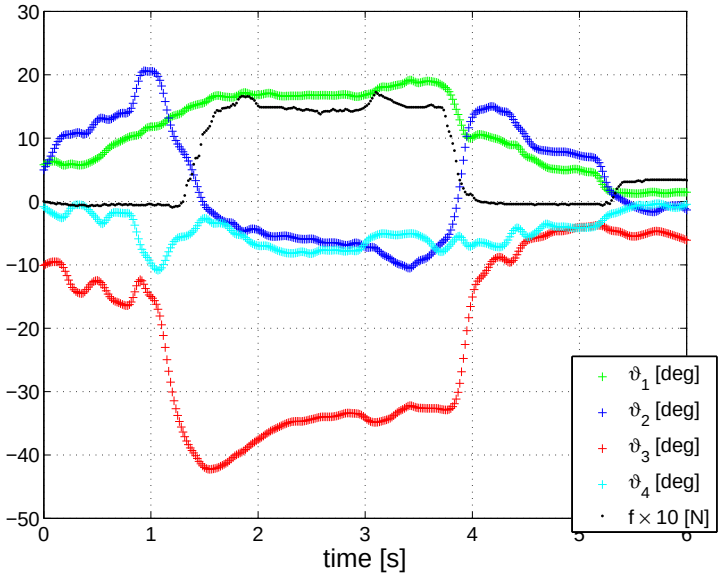


Figure 3.23: Index joint angles and estimated tactile forces for the bottle shown in Fig. 3.27 (c)

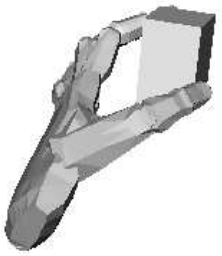


Figure 3.24: Example of visualization of the hand during a grasp of the parallelepiped shown in Fig. 3.26 (a). The hand is in a pre-grasp posture.

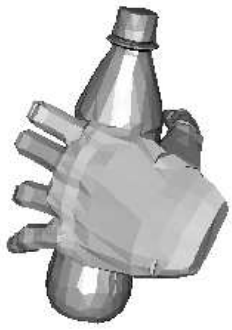


Figure 3.25: Example of visualization for a grasp of the empty plastic bottle in Fig. 3.26 (b)

problems such as Programming by Demonstration, object recognition, segmentation of elementary actions, activity recognition, low-level trajectory generation and other learning issues, by using not only the kinematic data, but also contact force information.

The number of the objects has been kept intentionally low, since each objects is representative of an entire class. For example a can represents all the cylindric-shaped object involved in typical manipulation tasks. Figures 3.19, 3.21, 3.22, 3.20, 3.18, 3.23 show examples of the estimated joint angles and the measured contact force for the index finger. Figures 3.24 and 3.25 are screenshots taken during the visual reproduction of the observed tasks. Some of the objects involved are shown in Fig. 3.26 and Fig. 3.27

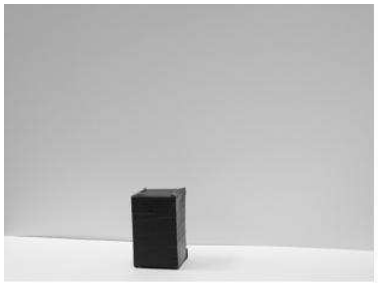



Figure 3.26: (a) parallelepiped, (b) plastic bottle



Figure 3.27: (a) little cup, (b) small bottle, (c) big bottle

3.3.2 Object Description

Parallelepiped (Fig. 3.26a)

weight: 0.016 kg

height: 70 mm

width: 40 mm

Filled plastic bottle (Fig. 3.26b)

weight: 0.35 kg

base diameter: 60 mm

height: 200 mm

Empty plastic bottle (Fig. 3.26b)

weight: 0.030 kg

base diameter: 60 mm

height: 200 mm

Little Cup (Fig. 3.27a)

weight: 0.010 kg

base diameter: 45 mm

handle diameter: 25 mm

height: 65 mm

Small bottle (Fig. 3.27b)

weight: 0.20 kg

base diameter: 55 mm

height: 220 mm

Big Bottle (Fig. 3.27c)

weight: 0.40 kg

base diameter: 70 mm

height: 210 mm

3.3.3 Structure of the captured data

A MATLAB nested structure is associated with each trial. The structure contains information about time, hand pose in space, kinematic model of the performer's hand and, for each finger, joint angles and fingertip contact forces. In detail, the structure is organized as follows:

thumb: it contains information about angles, contact forces and kinematic parameters of the thumb
(type: *structure*)

thumb.angles: it contains information about thumb joint angles
(type: *structure*)

thumb.angles.values: estimated values of the thumb joint angles. The column i contains the values of the angle ϑ_i (see table 3.5).
(type: $m \times 4$ *double matrix*) [2]

thumb.angles.meas_unit: measurement unit of the joint angles.
(type: *string*)

thumb.tactile: it contains information about the thumb contact force.
(type: *structure*)

thumb.tactile.values: normal component of the thumb contact force.
(type: $m \times 1$ *double matrix*)

thumb.tactile.meas_unit: measurement unit of the contact force.
(type: *string*)

thumb.dh: it contains the Denavit-Hartemberg (D-H) parameters of the thumb, that is a_2, a_3, a_4 (see table 3.5).
(type: *structure*)

thumb.dh.values: values of the D-H parameters a_2, a_3, a_4 (see table 3.5).
(type: 1×3 *double matrix*)

thumb.dh.meas_unit: measurement unit of the thumb D-H parameters a_2, a_3, a_4 (see table 3.5).
(type: *string*)

thumb.base: it contains the transformation matrix $\boldsymbol{T}_w^0$, where $\boldsymbol{O}_w$-$\boldsymbol{x}_w\boldsymbol{y}_w\boldsymbol{z}_w$ is a frame fixed on the hand palm and $\boldsymbol{O}_0$-$\boldsymbol{x}_0\boldsymbol{y}_0\boldsymbol{z}_0$ is defined according to D-H convention, see [20] for more detail.
(type: *structure*)

thumb.base.values: numerical value of $\boldsymbol{T}_w^0$ [20].
(type: 4×4 *double matrix*)

thumb.base.tr_meas_unit measurement unit of the translational part of the matrix $\boldsymbol{T}_w^0$.
(type: *string*)

index: see the description of **thumb**.

medium: see the description of **thumb**.

ring: see the description of **thumb**.

[2] m is the number of frames

link	a	d	ϑ	α
1	0	0	ϑ_1	$\frac{\pi}{2}$
2	a_2	0	ϑ_2	0
3	a_3	0	ϑ_3	0
4	a_4	0	ϑ_4	0

Table 3.5: D-H table for all the fingers.

little: see the description of **thumb**.

hand_position: information about the hand position with respect to the world frame $\boldsymbol{O}_b$-$\boldsymbol{x}_b\boldsymbol{y}_b\boldsymbol{z}_b$.
(type: *structure*)

hand_position.values: hand position in the base frame.
(type: $m \times 3$ *double matrix*)

hand_position.meas_unit: measurement unit of hand position with respect to the base frame.

hand_orientation: information about the hand orientation with respect to the world frame $\boldsymbol{O}_b$-$\boldsymbol{x}_b\boldsymbol{y}_b\boldsymbol{z}_b$.
(type: *structure*)

hand_orientation.values: hand orientation with respect to the base frame expressed in roll-pitch-yaw angles.
(type: $m \times 3$ *double matrix*)

hand_orientation.meas_unit: measurement unit of hand orientation angles with respect to the base frame.

time: information about the acquisition time.
(type: *structure*)

time.values: numerical values of the acquisition time.
(type: $m \times 1$ *double matrix*)

time.meas_unit: time measurement unit.
(type: *string*)

Chapter 4

High-level sensor fusion

The high-level sensor fusion module was initially developed to improve the rough hand observation system at the Humanoids and Intelligence Systems Laboratory of Karlsruhe Institute of Technology (KIT). Afterwards, it has been integrated without extra effort within the architecture described in Chapter 1. It has been possible since the high-level module only need a low-level measurement system that estimates the hand posture and the measurement contact forces and is completely independent from the particular technology adopted by the lower level.

The sensory environment at KIT, see Fig. 4.1, consists of a dataglove, a magnetic field tracker, a glove with tactile sensors and a stereo vision system.

The Cyberglove II dataglove measures 22DOF of the human hand, four joint values of each finger and two joint values of the wrist. The sensor resolution is $< 1^\circ$ and repeatability is 3°. Each time the datagloves are put on the position of the sensors is slightly different and they have to be calibrated. The calibration is done using a gesture based system to measure the distance between the measured joint angles and the joint angles of a predefined hand configuration, which the human teacher has to reproduce.

The measured joint angles are mapped to a predefined model of the human hand. In general, two different kinds of hand models are available: user-independent, see Fig. 4.2(a), and user-dependent, see Fig. 4.2(b). The user-independent model was created manually. The user-dependent hand model, generated using a laser scanner, usually provides a better accuracy. However, in order to allow different persons to teach the robot system, the user-independent hand model was used for the experiments, which induces larger errors.

The glove with tactile sensors is based on the Pressure Profile FingerTPS, see Fig. 4.3. In each fingertip, a tactile sensor pad is used to measure the force intensity applied to an object. Due to the size and location of the

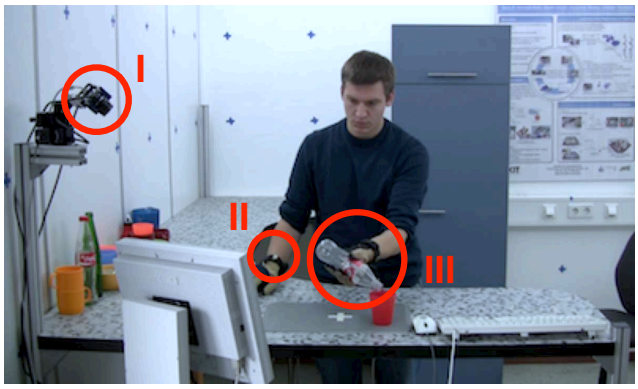
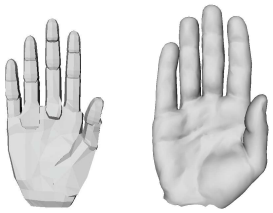


Figure 4.1: Sensor setup: DragonFly II stereo camera system (I), Polhemus Fastrak magnetic field based tracking device (II), Cyberglove II (III)



(a) User-independent (b) User-dependent

Figure 4.2: Hand models

Figure 4.3: Sensor setup II: Pressure Profile FingerTPS tactile sensors [4]

pads, the human operator has to consider the sensor pads and adapt the manipulation motion in order to make consistent force measurements. An additional wrist sensor is available but has not been used in this work. The system is calibrated using a provided dynamometer. The repeatability is $< 4\%$ of the full scale range, which is 4.55 to 22.73kg [4].

The position and orientation of the human wrist is measured using a Polhemus Fastrak magnetic field based tracking device. The static accuracy of the tracker is 0.8mm RMSE for the position and 0.15° RMSE for the orientation. In our lab environment, a different reference frame is used and the result may be distorted by metal objects like computers. After calibration, the position RMSE is 1.17mm and the orientation RMSE is 0.51° measured in a small area above the table using a chessboard.

The stereo vision system with two DragonFly II cameras is used to localize known objects in the environment. The IVT library [8] is trained with a 3D

object model to generate different views of the object, which are then used for the localization algorithm. The 3D models are available in the KIT ObjectModels Web Database [47].

Multiple noise sources exist: the standardized kinematics and geometric model of the human hand, the object position and orientation, the tracker position and orientation and the transformation of the tracker pose to the pose of the geometric model, i.e. the pose of the tracking device on the dataglove. As a consequence of this, the fingertip pose estimates relative to the object are inaccurate and inconsistencies between tactile measurements and fingertip position occur, e.g. a contact is measured by the tactile sensors but the 3D model of the human fingertip has no contact with the surface of the 3D object model.

4.1 Correction method

The observation phase including the proposed correction algorithm is constituted by three steps:

1. data acquisition and set up of the virtual environment that reproduce the "real" one
2. use of the geometric information and contact information to define the correction of the fingertip positions and orientations
3. use of an inverse kinematic algorithm, that finds the pose of the hand and the joint angles required to implement the correction

The virtual environment is constituted by the 3D models of the hand and of the objects involved in the observed task. In this work, three different objects have been considered: a cup, a small plate, a bottle. The hand is animated with the estimated pose and joint angles during the observation phase and the position of the object is obtained by the stereo vision system. Nevertheless, the proposed algorithm is absolutely general and can be adopted with every measurement system able to estimate the hand posture and the position of the object. With no measurement error and no 3D modeling error, the virtual environment reproduces exactly the real one. In this work, the errors in object 3D models are supposed to be much smaller than the measurement errors. These often cause inconsistencies between measured contact forces and hand posture relative to the object. In order to improve the accuracy of the observed data and specifically to make the motion data consistent with the contact force data, corrections of the hand pose and finger configurations have to be computed and implemented. The pseudo-code

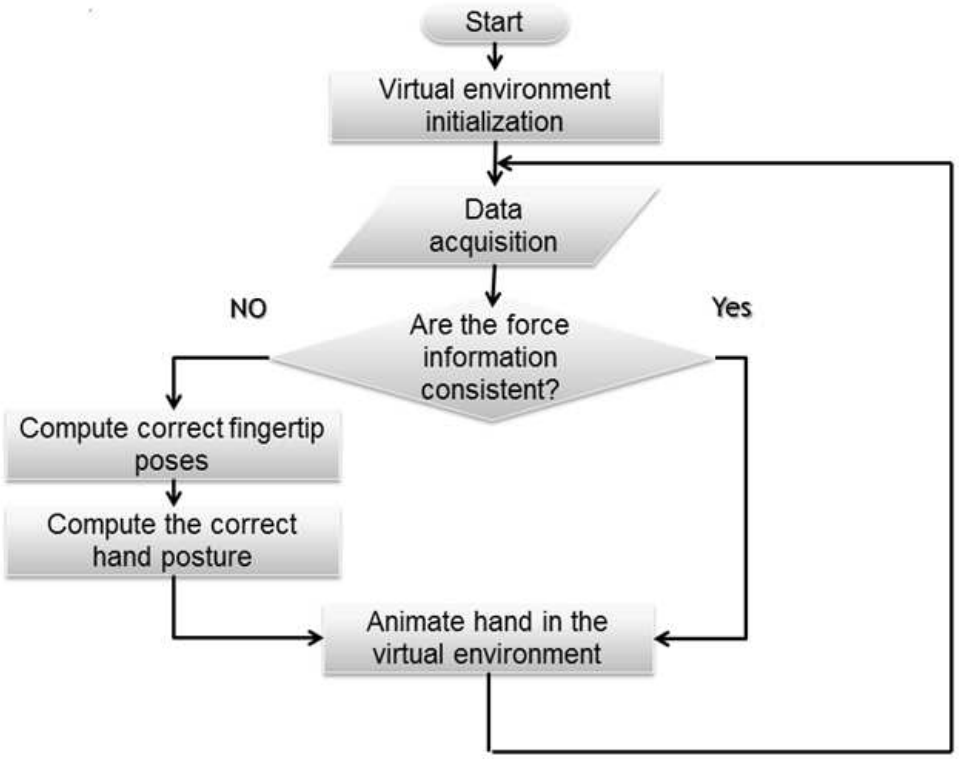


Figure 4.4: Flowchart of the correction algorithm used in the high-level sensor fusion component

of the correction method is described in Algorithm 1. When a measured fingertip contact force exceeds the empirical defined threshold, the finger is considered in contact with the object (line 5). In this work, the threshold has been fixed at 0.5 N. After the contact checking in the "real environment", in order to determine if a collision in the virtual environment occurs, the collision checker described in Section 4.2 is called for each finger (line 10). When the collision checker is called, for each finger f, it requires as input two 3D models (see Algorithm 2). The first 3D model, $\mathbf{CA}_f$, is the tactile sensor pad model, which is a thin plate modeling the expected contact area of the finger f. The second 3D model, $\boldsymbol{O}$ is the model of the grasped object.

If in the real world, according to the force measurement, a finger is considered in contact with the object and in the virtual environment no collision is detected or vice-versa (line 11), the finger is considered not-consistent and the correction is computed on the basis of the information provided by the

Algorithm 1 Correction algorithm

1: **for** each frame k **do**
2: $(\boldsymbol{q}_0, \textbf{force}) = \text{readSensorData}()$
3: $\boldsymbol{x}_{goal} = []$
4: **for** each finger f **do**
5: **if** $force_f > threshold$ **then**
6: $realContact_f = true$
7: **else**
8: $realContact_f = false$
9: **end if**
10: $\left(\hat{\boldsymbol{n}}_f, coll_f, d_f, \boldsymbol{p}_{0_f}\right) = \text{checkCollision}(\mathbf{CA}_\text{f}, \boldsymbol{O})$
11: **if** $coll_f \neq realContact_f$ **then**
12: $\boldsymbol{p}_f = \boldsymbol{p}_{0_f} + d_f \cdot \hat{\boldsymbol{n}}_f$ {correction required}
13: $\boldsymbol{x}_{goal} = \text{append}(\boldsymbol{x}_\text{goal}, [\boldsymbol{p}_\text{f}, \hat{\boldsymbol{n}}_\text{f}(1:2)])$
14: **end if**
15: $\boldsymbol{q}_{goal} = \text{CLIK}\left(\boldsymbol{x}_\text{goal}, \boldsymbol{q}_0, \boldsymbol{W}, \alpha\right)$
16: **end for**
17: $\text{moveHand}(\boldsymbol{q}_\text{goal})$
18: **end for**

collision checker, i.e. the versor $\hat{\boldsymbol{n}}$, the point $\boldsymbol{p}_0$ and the scalar d, defined in Section 4.2.

As explained in the Algorithm 2, these data allow finding the point $\boldsymbol{p}_f$ on the object $\boldsymbol{O}$, such that the distance between the pad model of the finger f and the object is minimum.

The object point $\boldsymbol{p}$ and two of three components of the versor $\hat{\boldsymbol{n}}$ are included in the vector $\boldsymbol{x}_{goal}$. Its elements specify the desired poses of the fingertips for the correction step.

In line 15, the vector $\boldsymbol{x}_{goal}$ is given as input to the CLIK algorithm (see Algorithm 3 in Section 4.3), which is in charge of finding the hand configuration $\boldsymbol{q}_{goal}$ such that, for each non-consistent finger, the fingertip position is $\boldsymbol{p}_f$ and the versor normal to the pad $\mathbf{CA}_f$ is aligned with the versor $\hat{\boldsymbol{n}}_f$. The starting configuration $\boldsymbol{q}_0$ of the CLIK algorithm is taken as the hand configuration measured by the sensor setup.

4.2 Geometric information from the scene

To obtain the geometric information necessary to define the correction of the fingertip poses, the collision checker Proximity Query Package (PQP) [50]

has been used in combination with Openrave [31]. Algorithm 2 describes the high level behavior of the collision checker, based on PQP. The input data of the collision checker module are two 3D models: $model_1$ and $model_2$.

Algorithm 2 Collision checker behaviour

1: $(\hat{\boldsymbol{n}}, coll, d, \boldsymbol{p}_0)$=checkCollision(model$_1$, model$_2$) :
2: $collision$ = isColliding$(model_1, model_2)$
3: **if** $collision == false$ **then**
4: $\quad \boldsymbol{p}_0$ = computePoint$(model_1, model_2)$
5: $\quad d$ = computeDistance$(model_1, model_2)$
6: $\quad \hat{\boldsymbol{n}}$ = computeNormal$(model_1, model_2)$
7: $\quad$ **return** $(\hat{\boldsymbol{n}}, collision, d, \boldsymbol{p}_0)$
8: **else**
9: $\quad \boldsymbol{p}_0$ = computePenetrationPoint$(model_1, model_2)$
10: $\quad d$ = computePenetrationDepth$(model_1, model_2)$
11: $\quad \hat{\boldsymbol{n}}$ = computeNormal $(model_1, model_2)$
12: $\quad$ **return** $(\hat{\boldsymbol{n}}, collision, d, \boldsymbol{p}_0)$
13: **end if**

The output variables $\boldsymbol{p}_0$, $\hat{\boldsymbol{n}}$, d are such that:

- When $model_1$ and $model_2$ are not in collision, the point $\boldsymbol{p} = \boldsymbol{p}_0 + d \cdot \hat{\boldsymbol{n}}$ is the point on $model_2$ closest to $model_1$, d is the minimum distance between $model_1$ and $model_2$, $\boldsymbol{p}_0$ is the point on $model_1$ closest to $model_2$, $\hat{\boldsymbol{n}}$ is the versor of the line connecting $\boldsymbol{p}$ and $\boldsymbol{p}_0$.

- When $model_1$ and $model_2$ are in collision, the point $\boldsymbol{p} = \boldsymbol{p}_0 + d \cdot \hat{\boldsymbol{n}}$ is the point on the border of $model_2$ farthest from the part of $model_1$ in collision with $model_2$. In this case it is $d < 0$. Briefly, if the translation $\boldsymbol{p} - \boldsymbol{p}_0$ is applied, $model_1$ is brought outside $model_2$ and it is $d = 0$.

Since, for each non-consistent finger f, the point $\boldsymbol{p}_f$ and the versor $\hat{\boldsymbol{n}}_f$ will be set as corrected positions and orientations in the task space, the correction is computed according to a minimum distance criterion.

4.3 Inverse kinematics method

Algorithm 3 explains the pseudo-code for the inverse kinematics. Line 1 is the definition of the function, which specifies the input and output variables.

The task space goal $\boldsymbol{x}_{goal}$ contains, for each not-consistent finger, the desired position and orientation, computed by calling the collision checker.

For a finger, the position is the point $\boldsymbol{p}$ and the orientation is represented by the first two components of the vector $\hat{\boldsymbol{n}}$, defined in Section 4.2.

The authors have empirically observed that the performance in terms of velocity and stability is often better, if the orientation is specified only for one or two fingers. In fact, when the size of Jacobian matrix increases too much, a significant decreasing in the performance may occur. Hence, in all the experiments presented in Section 4.5, the orientation has been fixed only for the first two not-consistent fingers.

The vector $\boldsymbol{q}_0$ contains the measured hand posture, i.e. position, orientation and joint angles. It is the starting point of the IK algorithm. The more accurate the measurement system is, the closer $\boldsymbol{q}_0$ is to the correct minimum and then the faster and more effective the search is.

To take into account the differences in variances of the sensors and in magnitude order, the weighted pseudo-inverse can be computed instead of the simple right pseudo-inverse.

$$\boldsymbol{J}^{\dagger} = \boldsymbol{W}^{-1}\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{\mathrm{T}})^{-1} \tag{4.1}$$

where the weight matrix $\boldsymbol{W}$ can be chosen in this form:

$$\boldsymbol{W} = f(\boldsymbol{\Sigma}) \tag{4.2}$$

with f monotonically increasing function of the sensor standard deviations $\sigma_1, \sigma_2, ..., \sigma_n$ and

$$\boldsymbol{\Sigma} = \mathrm{diag}\left(\sigma_1, \sigma_2, ..., \sigma_{\mathrm{n}}\right). \tag{4.3}$$

The simplest choice for $\boldsymbol{W}$ is the following:

$$\boldsymbol{W} = w\boldsymbol{\Sigma} \tag{4.4}$$

where w is a positive scalar.

Algorithm 3 Inverse Kinematic Algorithm

$\boldsymbol{q}_{goal} = \mathrm{CLIK}\left(\boldsymbol{x}_{\mathrm{goal}}, \boldsymbol{q}_0, \boldsymbol{W}, \alpha, \boldsymbol{k}(.)\right)$:
$\boldsymbol{e} = \boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q}_0)$
while $\|\boldsymbol{e}\| > \varepsilon$ **do**
  $\boldsymbol{J}(\boldsymbol{q}) = \mathrm{computeJacobian}\left(\boldsymbol{k}(.), \boldsymbol{q}\right)$
  $\boldsymbol{q} = \boldsymbol{q} + \alpha\boldsymbol{J}^{\dagger}(\boldsymbol{q})(\boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q}))$
  $\boldsymbol{e} = \boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q})$
end while
return $\boldsymbol{q}_h$

With this choice, the algorithm will tend to modify more the variables whose measurement are less reliable and less the variable whose measurements are more reliable. On the other hand, when the magnitude order of the sensor variance is too different, a possible choice is

$$\boldsymbol{W} = \boldsymbol{K}\boldsymbol{\Sigma} \tag{4.5}$$

$$\boldsymbol{K} = \text{diag}\left(k_1, k_2, ..., k_n\right). \tag{4.6}$$

With this choice, the parameters $k_1, k_2, ..., k_n$ have usually to be fixed empirically.

The scalar α is the so-called CLIK gain. The gain strongly influences the speed of convergence and the stability of the algorithm. In [36], a theoretical study on the stability of closed loop inverse kinematics algorithms is proposed and it can be an useful tool to correctly tune the parameters of the IK algorithm and, in particular, the gain α.

The last input parameter is $\boldsymbol{k}(.)$, the direct kinematics function. In this application, k is a function of the position of the hand $\boldsymbol{p}_{hand}$, the orientation of the hand $\boldsymbol{\Phi}_{hand}$ and the finger joint angles $\boldsymbol{q}_{joint}$. It is defined as: $\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$, $\boldsymbol{q} = \begin{bmatrix}\boldsymbol{p}_{hand} & \boldsymbol{\Phi}_{hand} & \boldsymbol{q}_{joint}\end{bmatrix}^T$ and $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}_{pos} & \boldsymbol{x}_{orientation}\end{bmatrix}^T$. The main advantage to have as input the direct kinematics function is that the algorithm does not require any changes if a different hand kinematic model is adopted. For this reason, the Jacobian is also numerically evaluated (line 4). Since the Jacobian computation is much faster than the pseudo-inverse computation, the Jacobian numerical evaluation does not cause a significant delay in finding the IK solution.

4.4 Collision-free correction

The basic version of the algorithm looks for a posture of the hand that guarantees the consistency between posture and fingertip contact forces. Nevertheless, for some tasks, collisions might occur between parts of hand (e.g. palm) and the grasped object. An example of collision is shown in Fig. 4.6, in which the hand palm is in collision with the cup. To handle this problem, a modified version of the CLIK algorithm has been developed that, in case of collisions, computes the corrected hand posture considering two tasks. The first task consists of finding a hand posture such that the fingertips are in the desired poses, while the second task consist of finding a collision-free hand posture. A priority level is associated with each task. In this work, the accurate placement of the fingertips in the desired poses has the high-level priority since it guarantees the kinetostatic consistency, very important in PbD applications. Classical techniques to provide a system with the possibility of

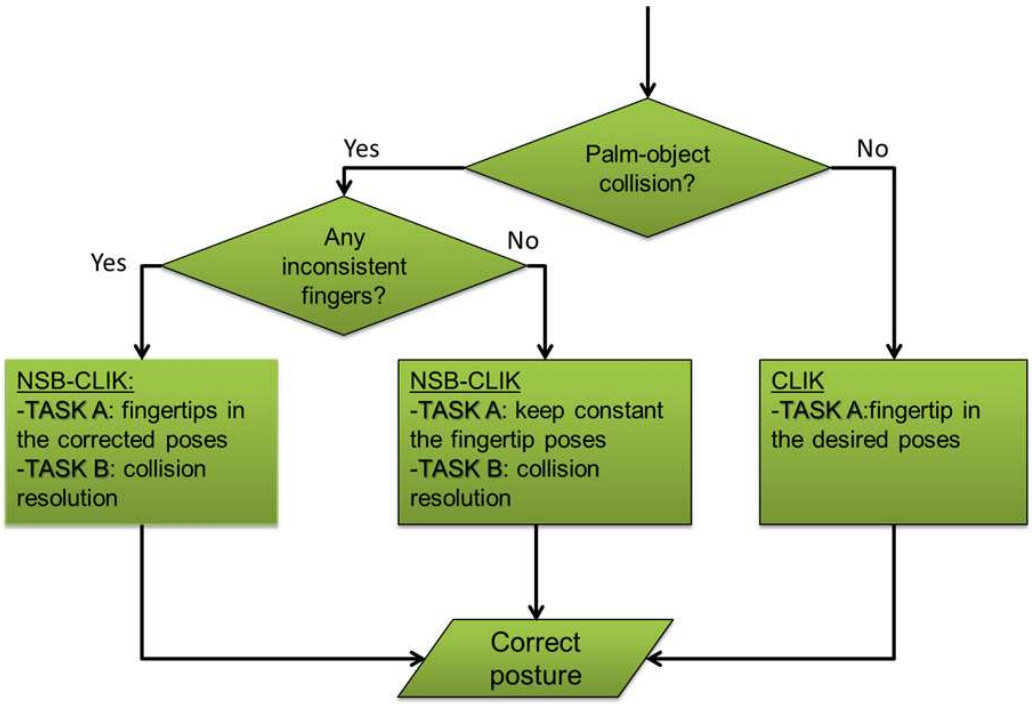


Figure 4.5: Flowchart of the extended CLIK algorithm for a collision-free posture correction

executing multiple tasks are the so called behavior-based approaches. The key idea is that the intelligence of the system is provided by a set of behaviors (tasks in this case), designed to achieve specific goals, that are activated on the basis of external information. One of the most adopted approach in robotics is the subsumption architecture, where each task is related to a layer that is an asynchronous module communicating with the others. Layers have different priority levels, and the possible conflict among the tasks is solved by assigning a hierarchy so that the higher-level task can subsume the lower-levels.

Another behavior-based approach, called Null Space Based (NSB), is described in [7] for mobile robotics applications. In the NSB approach, behaviors are merged to define the final motion directives to the robots. In particular, the behaviors are arranged in priorities, and they are composed using null-space projection matrices so that multiple behaviors are simultaneously activated but the lower-priority behaviors do not affect the higher-priority ones. In fact, the NSB always fulfils the highest-priority task; the lower-priority tasks, on the other hand, are fulfilled only in a subspace where they do not conflict with the ones having higher priority. The NSB approach is

Algorithm 4 NSB Inverse Kinematic Algorithm

$\boldsymbol{q}_{goal} = \text{NSBCLIK}(\boldsymbol{x}_{goal}, \boldsymbol{q}_0, \boldsymbol{W}, \alpha, \boldsymbol{k}(.), d_{goal})$:
$\boldsymbol{e} = \boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q}_0)$
while $\|\boldsymbol{e}\| > \varepsilon$ **do**
  $\boldsymbol{J}(\boldsymbol{q}) = \text{computeJacobian}\,(\boldsymbol{k}(.), \boldsymbol{q})$
  $(\hat{\boldsymbol{n}}, d) = \text{checkCollision}(\boldsymbol{O}, \textbf{palm})$
  $\boldsymbol{q} = \boldsymbol{q} + \alpha \boldsymbol{J}^{\dagger}(\boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q})) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})(\hat{\boldsymbol{n}}\,(d_{goal} - d))$
  $\boldsymbol{e} = \boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q})$
end while
return $\boldsymbol{q}_h$

particularly suitable for the application described in this work, since it can be derived by simply extending the CLIK algorithm described in Section 4.3.

Eq. 4.7 represents the core of the NSB CLIK algorithm, that is executed when a collision occurs before the correction or after the correction with the basic CLIK; it is constituted by two terms. The first term represents the high-priority task, in this case it consists of finding a posture that brings the fingertips to the desired poses and the second term is the low-priority task that brings the hand in a collision-free posture. The low-level priority task is projected into the null-space of the the high-priority task through the projector $(\boldsymbol{I} - \boldsymbol{J}_k^{\dagger}\boldsymbol{J}_k)$. As a consequence, the algorithm will converge, among the infinite solutions satisfying the high-priority task, to a solution such that also the low-level priority task is fullfilled, if such a solution exists.

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \alpha \boldsymbol{J}_k^{\dagger}(\boldsymbol{x}_{goal} - \boldsymbol{k}(\boldsymbol{q}_k)) + (\boldsymbol{I} - \boldsymbol{J}_k^{\dagger}\boldsymbol{J}_k)(\boldsymbol{J}_{c_k}^{\dagger}\,(d_{goal} - d_k)) \tag{4.7}$$

The matrix $\boldsymbol{J}_c$ is defined as

$$\boldsymbol{J}_c = \frac{\partial d}{\partial \boldsymbol{p}} = \frac{\partial \,\|\boldsymbol{p} - \boldsymbol{p}_0\|}{\partial \boldsymbol{p}} = \frac{\boldsymbol{p}^T - \boldsymbol{p}_0^T}{\|\boldsymbol{p} - \boldsymbol{p}_0\|} = \hat{\boldsymbol{n}}^T$$

and, according to the definition of pseudo-inverse, it is:

$$\boldsymbol{J}_c^{\dagger} = \hat{\boldsymbol{n}}.$$

The unit vector $\hat{\boldsymbol{n}}$, the scalar d and the points $\boldsymbol{p}_0$ are outputs of the collision checker and have been defined in Section 4.2. Algorithm 4.4 describes in more details the NSB extension of CLIK method and the flow-chart in Fig 4.5 shows the strategy for the collision-free correction. Algorithm 4.4 considers as an example only palm-object collisions, but it is worth to note that the same method can be adopted to find collision-free posture with each part

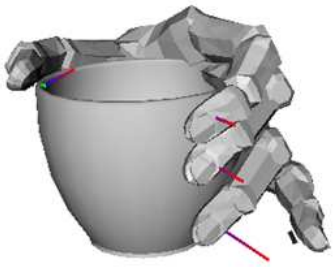


Figure 4.6: Example of palm-object collision

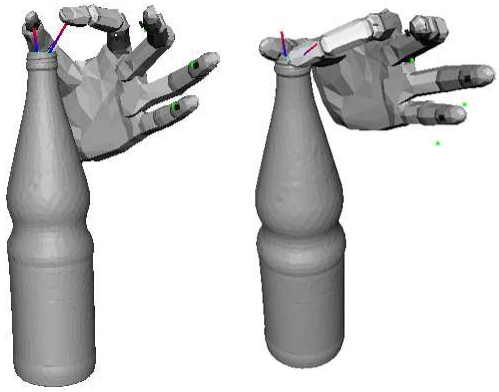


Figure 4.7: Initial posture (left), corrected posture (right)

of the hand or also selfcollision-free hand postures. The obvious drawback is that the more collision avoidance features are included in the NSB-CLIK the slower is the execution time of the algorithm itself. In fact the main drawback of Algorithm is that the collision checker is called in each iteration. As a consequence, with relatively slow computer systems it may not work on-line effectively.

4.5 Results and discussion

To test the performance of the developed algorithm, a set of experiments have been performed, consisting of three different tasks: unscrewing a bottle, grasping a cup and grasping a plate. For each "observed" task, the frame by frame consistency analysis and correction have been carried out. In all the frames of each experiment, the consistency condition has been successfully attained. In this section, the results concerning the two or three frames per experiment, which require the highest correction effort, are discussed. The

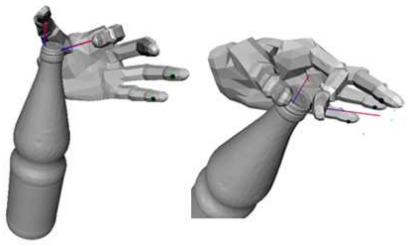


Figure 4.8: Initial posture (left), corrected posture (right)

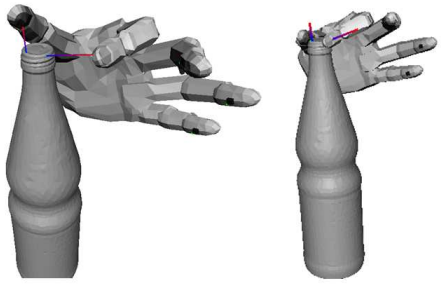


Figure 4.9: Initial posture (left), corrected posture (right)

results are summarized in the Figures 4.7-4.11 and in the Tables 4.1, 4.3, 4.4. In Figures 4.7, 4.8, 4.9 some frames of the "Unscrewing a bottle" task are shown. In the left part of the three figures, according to the tactile measurements (see Algorithm 1), the index and thumb fingers should be in contact with the bottle. Nevertheless, the collision checker of the virtual environment computes no contact. Hence, a correction (shown on the right part) has been applied to the pose of the hand and to the thumb and index joint angles.

Figures 4.10 and 4.11 show the task "grasping a plate". In both frames, the thumb and ring fingers have no contacts with the object, but the measured fingertip forces exceed the threshold. As a consequence, the consistency condition is not satisfied and a correction is applied. In Fig. 4.13 and 4.12 the task "grasping a cup" is represented. It is interesting to note that in Fig. 4.13 a collision occurs between the cup and the proximal phalange of the middle finger. It may happen, when the initial error norm is too large. At the moment, to allow the on-line execution, the collision avoidance is not included in the algorithm functionalities. If for the specific application it is important to avoid collisions, Algorithm 4.4 has to be adopted in the computing of the hand posture. Yet, with a good sensor setup, the collision phenomena are quite rare. Table 4.1 shows the initial and corrected hand

Fig.	Initial Position [m]	Corrected Position [m]
4.7	$(0.955, -0.07, 1.06)$	$(0.907, 0.030, 1.15)$
4.8	$(0.952, -0.07, 1.03)$	$(0.902, 0.028, 1.14)$
4.9	$(0.958, -0.07, 1.05)$	$(0.879, 0.022, 1.15)$
4.12	$(0.884, -0.074, 0.905)$	$(0.802, -0.046, 0.934)$
4.13	$(0.778, -0.008, 0.950)$	$(0.818, -0.021, 0.929)$
4.10	$(0.895, -0.0008, 0.905)$	$(0.817, 0.0068, 0.987)$
4.11	$(0.894, -0.0007, 0.905)$	$(0.820, 0.0010, 0.990)$

Table 4.1: Initial and corrected position of the hand

position and Table 4.3 shows the initial and corrected finger joint angles. The qualitative improvement of the hand configuration estimation can be appreciated from the presented screenshots, moreover the obtained coherence of the sensor data will allow to improve the following learning phase within the PbD framework.

In Table 4.4, some parameters are listed to evaluate the performance of the IK algorithm. With a careful choice of the tolerance ϵ and the gain α, only few iterations are required for the algorithm to converge. To have a rough idea of the execution time, a single pseudo-inverse computation of the CLIK algorithm has required, for two not-consistent fingers, an average computation time of $1.5\,\mu$s, evaluated on 1000 trials in a C++ implementation, running on a Intel Core2Duo with 4GB RAM. The execution of the collision checking algorithm for a single fingertip takes on average 0.5 ms.

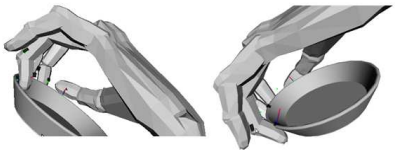


Figure 4.10: Initial posture (left) and corrected posture (right)

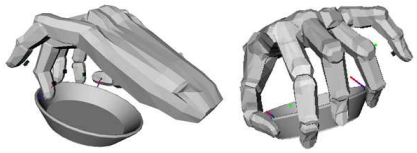


Figure 4.11: Initial posture (left), corrected posture (right)

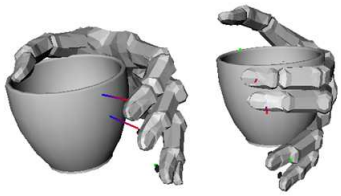


Figure 4.12: Initial posture (left), corrected posture (right)

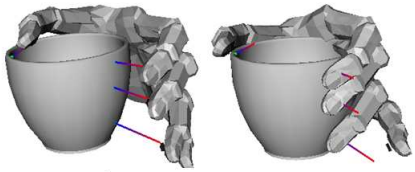


Figure 4.13: Initial posture (left), corrected posture (right)

Fig	Init Orientation [rad]	Corr. Orientation [rad]
4.7	$(2.86, -0.987, 1.56)$	$(2.84, -0.987, 1.56)$
4.8	$(2.93, -0.966, 1.58)$	$(2.94, -0.966, 1.57)$
4.9	$(3.08, -1.05, 1.45)$	$(3.09, -1.05, 1.45)$
4.12	$(-2.18, -1.21, 0.195)$	$(-1.52, -1.69, -0.017)$
4.13	$(-1.49, -1.04, -0.292)$	$(-1.52, -1.20, -0.400)$
4.10	$(-1.54, -0.0454, 0.244)$	$(-1.45, 0.0553, -0.171)$
4.11	$(-1.56, -0.043, 0.295)$	$(-1.49, 0.045, -0.133)$

Table 4.2: Initial and corrected orientation expressed in ZYZ Euler angles

F	Fig.	Initial Angles [rad]	Corrected Angles [rad]
T	4.7	$(.93, -.96, -.37, -.48)$	$(1.54, -.93, -.052, .06)$
I		$(-.74, -.065, -.60, -.26)$	$(-1.01, .1, -1.08, .22)$
T	4.8	$(.91, -1.1, -.19, -.53)$	$(1.42 - .99, .011, .006)$
I		$(-.85, -.057, -.48, -.26)$	$(-1.07, 0.005, -.88, .20)$
T	4.9	$(.96, -1.04, -0.18, -.53)$	$(1.46, -.86, -.12, -.13)$
I		$(-.61, -.062, -.84, .03)$	$(-.71, .16, -1.38, .41)$
T	4.12	$(.9, -1.1, -.49, -.45)$	$(.9, -1.07, -.5, -.45)$
I		$(-.19, -.02, -.76, -.42)$	$(-.42, .01, -.82, -.56)$
M		$(-.29, 0, -.7, -.11,)$	$(-.35, -.07, -.92, -.24)$
R		$(-.36, .19, -.49, -.07)$	$(-.37, .19, -.48, -.067)$
I	4.13	$(-.33, .25, -.9, -.79)$	$(-.32, .31, -.87, -.8)$
M		$(-.51, .13, -.84, -.15)$	$(-.51, .16, -.88, -.17)$
T	4.10	$(.9, -.86, -.41, -.70)$	$(.73, -.79, -.39, -.72)$
R		$(-.78, .21, -.98, -.69)$	$(-.76, .21, -.90, -.66)$
M	4.10	$(-.78, 0, -1.1, -.15)$	$(-.59, -.13 - 1.17, -.23)$
R		$(-.92, .24, -.9, -.7)$	$(-.93, .24, -.81, -.65)$
L		$(-.76, .49, -.86, -.15)$	$(-.77, .48, -.87, -.15)$

Table 4.3: Initial and corrected finger joint angles. T=Thumb, I=Index, M=Middle, R=Ring, L=Little

Fig.	$\|\boldsymbol{e}_0\|$	α	N.o.I.	ϵ
4.7	0.311	0.8	8	10^{-6}
4.8	0.243	0.8	9	10^{-6}
4.9	0.318	0.8	9	10^{-6}
4.12	0.2691	1.1	6	10^{-6}
4.13	0.281	1.1	6	10^{-6}
4.10	0.19	1.1	6	10^{-6}
4.11	0.20	1.1	6	10^{-6}

Table 4.4: Initial error norm, gain, number of iterations and tolerance of the IK algorithm

4.6 A stability theorem to tune the CLIK gain

This Section presents novel proofs of convergence of two kinds of CLIK algorithms directly in the discrete-time domain, thus leading to useful guidelines for gain selection in relation to the sampling time. The adopted methodology is not based on Lyapunov arguments, which are not straightforward to apply in the case of redundant robots and sometimes easily lead to technical errors. In fact, in such a case proving that the origin of the task space error space is asymptotically stable is not so trivial, since the Lyapunov function candidate depend not only on the task space but on the configuration variables too. Therefore, it cannot be shown to be positive definite without including terms depending on the configuration variables. The alternative approach followed here completely avoids the use of Lyapunov functions, nevertheless it rigorously proofs the stability of the algorithm according to the comparison principle for discrete-time systems. The presented simulation shows how the sufficient conditions for the jacobian pseudo-inverse algorithm are not conservative at all.

4.6.1 The inverse kinematics problem

Let $\boldsymbol{x} \in X$ be the vector of task variables of a robotic system with X a domain of $\mathbb{R}^m$ and let $\boldsymbol{q} \in Q$ be the vector of the robotic system configuration with Q a domain of $\mathbb{R}^n$, with $m \leq n$. For example, in a robotic manipulator $\boldsymbol{x}$ is the pose of the end effector and $\boldsymbol{q}$ is the vector of joint positions, whereas in a platoon of mobile robots, $\boldsymbol{q}$ is the vector of coordinates representing the location of each robot and $\boldsymbol{x}$ is the vector of suitable task variables depending on the mission.

The direct kinematics equation can be always written in the form

$$\boldsymbol{k} : Q \subseteq \mathbb{R}^n \rightarrow X \subseteq \mathbb{R}^m, \quad \boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}). \tag{4.8}$$

This function will be hereafter called "direct kinematics function" or "task function" without distinction.

Under the assumption of absence of kinematic singularities within the configuration space Q, the jacobian

$$\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{m \times n} \tag{4.9}$$

is full-rank $\forall\, \boldsymbol{q} \in Q$. This matrix will be hereafter called "robot jacobian" or "task jacobian" without distinction. Both the direct kinematics function and the robot jacobian are assumed to fulfill the following assumptions:

i) $\exists\, \delta > 0 :\ \|\boldsymbol{J}(\boldsymbol{q})\| \leq \delta,\ \forall\, \boldsymbol{q} \in Q$

ii) $\exists\, \beta > 0 :\ \underline{\sigma}(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})) \geq \beta,\ \forall\, \boldsymbol{q} \in Q$

iii) the function $\boldsymbol{k}(\boldsymbol{q})$ is smooth enough such that

$$\boldsymbol{k}(\boldsymbol{q} + \tilde{\boldsymbol{q}}) = \boldsymbol{k}(\boldsymbol{q}) + \boldsymbol{J}(\boldsymbol{q})\tilde{\boldsymbol{q}} + \boldsymbol{r}_k(\boldsymbol{q}),$$

where the reminder $\boldsymbol{r}_k(\boldsymbol{q})$ is such that

$$\exists\, \nu_k > 0 :\ \|\boldsymbol{r}_k(\boldsymbol{q})\| \leq \nu_k \|\tilde{\boldsymbol{q}}\|^2,\ \forall\, \tilde{\boldsymbol{q}}\ : \boldsymbol{q} + \tilde{\boldsymbol{q}} \in Q,$$

where, as matrix norm, the spectral norm, i.e. the largest singular value, has been assumed and the symbol $\underline{\sigma}(\boldsymbol{X})$ denotes the smallest singular value of the matrix $\boldsymbol{X}$. Assumptions *i)* and *ii)* are aimed at quantifying the distance from singularities, while assumption *iii)* is aimed at taking into account the degree of smoothness of $\boldsymbol{k}(\boldsymbol{q})$. Basically, the nonlinear functions have to possess second-order derivatives bounded, and fortunately in many applications such a requirement is verified. For example, every robot with revolute joints has a direct kinematics function constituted by polynomial combinations of trigonometric functions of the joint variables, therefore their second-order derivatives are certainly bounded. Such a degree of smoothness can be easily quantified by applying the following lemma.

Lemma 1. *Given a vector function defined in a domain* $D \subseteq \mathbb{R}^n$, $\boldsymbol{f} : \boldsymbol{x} \in D \to \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$ *with the Hessian matrices* $\boldsymbol{H}(f_i)$ *of all its components* $f_i(\boldsymbol{x}),\ i = 1, \ldots, m$ *norm-bounded uniformly in* D*, i.e.*

$$\exists\, \nu_i > 0 :\ \|\boldsymbol{H}(f_i(\boldsymbol{x}))\| \leq \nu_i,\ \forall\, \boldsymbol{x} \in D, \quad i = 1, \ldots, m \tag{4.10}$$

then, $\forall\, \tilde{\boldsymbol{x}} \in D :\ \boldsymbol{x} + \tilde{\boldsymbol{x}} \in D$ *and the whole line from* $\boldsymbol{x}$ *to* $\tilde{\boldsymbol{x}}$ *belongs to* D*, it is*

$$\boldsymbol{f}(\boldsymbol{x} + \tilde{\boldsymbol{x}}) = \boldsymbol{f}(\boldsymbol{x}) + \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \boldsymbol{r}_f(\boldsymbol{x}) \tag{4.11}$$

and the reminder $\boldsymbol{r}_f(\boldsymbol{x})$ *is such that*

$$\exists\, \nu_f > 0 :\ \|\boldsymbol{r}_f(\boldsymbol{x})\| \leq \nu_f \|\tilde{\boldsymbol{x}}\|^2 \tag{4.12}$$

Proof. The proof is a direct consequence of Taylor's theorem for functions of several variables [49] written with a second order reminder. This theorem

ensures that it always exists a point $\boldsymbol{\xi}$ on the line connecting $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, if this entirely belongs to the domain D, such that

$$\boldsymbol{f}(\boldsymbol{x}+\tilde{\boldsymbol{x}}) = \boldsymbol{f}(\boldsymbol{x}) + \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{1}{2}\begin{pmatrix} \tilde{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{H}(f_1(\boldsymbol{\xi}))\tilde{\boldsymbol{x}} \\ \tilde{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{H}(f_2(\boldsymbol{\xi}))\tilde{\boldsymbol{x}} \\ \vdots \\ \tilde{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{H}(f_m(\boldsymbol{\xi}))\tilde{\boldsymbol{x}} \end{pmatrix}.$$

Denoting the last term of the right-hand side with the symbol $\boldsymbol{r}_f$, this can be easily upper bounded as

$$\begin{aligned} \|\boldsymbol{r}_f(\boldsymbol{x})\| &\leq \sqrt{m}\|\boldsymbol{r}_f(\boldsymbol{x})\|_\infty = \sqrt{m}/2 \max_i |\tilde{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{H}(f_i(\boldsymbol{\xi}))\tilde{\boldsymbol{x}}| \leq \\ &\leq \sqrt{m}/2 \max_i \|\boldsymbol{H}(f_i(\boldsymbol{\xi}))\|\|\tilde{\boldsymbol{x}}\|^2 \leq \\ &\leq \sqrt{m}/2 \max_i \nu_i \|\tilde{\boldsymbol{x}}\|^2 \triangleq \nu_f \|\tilde{\boldsymbol{x}}\|^2 \end{aligned}$$

□

Let $\boldsymbol{x}_{d_h} \in X$, being $h \in \mathbb{Z}$ the discrete-time variable, be a desired task space position, the objective of any IK algorithm is to find one of the, in general many, configurations $\boldsymbol{q}_{d_h}$ such that

$$\boldsymbol{x}_{d_h} = \boldsymbol{k}(\boldsymbol{q}_{d_h}), \tag{4.13}$$

which is a system of m nonlinear equations, therefore the first ideas to tackle the problem resorted to iterative algorithms devoted to find zeros of nonlinear functions, e.g. the Newton-Raphson method in [10], [39]. In the following, two versions of the so-called CLIK algorithm are recalled [65], whose stability will be later proved in the discrete-time domain together with an estimation of the region of attraction.

Differently from analytic solutions (available only in special cases) and the iterative algorithms mentioned above, the CLIK algorithms rely on the inversion of the differential kinematics in the continuous-time domain, i.e.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{J}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t), \tag{4.14}$$

done in a closed-loop fashion as originally proposed in [65]

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{J}^{\dagger}(\boldsymbol{q}(t))\big(\dot{\boldsymbol{x}}_d(t) + \gamma(\boldsymbol{x}_d(t) - \boldsymbol{k}(\boldsymbol{q}(t))\big) \tag{4.15}$$

so as to avoid drift of the tracking error when implemented in discrete-time, e.g. by resorting to the Euler integration method with sampling time T, namely

$$\boldsymbol{q}_{h+1} = \boldsymbol{q}_h + T\boldsymbol{J}^{\dagger}(\boldsymbol{q}_h)(\dot{\boldsymbol{x}}_{d_h} + \gamma(\boldsymbol{x}_{d_h} - \boldsymbol{k}(\boldsymbol{q}_h)) \tag{4.16}$$

and to improve convergence rate acting on the positive gain γ. In the two equations above the symbol $\boldsymbol{X}^\dagger$ denotes the Moore-Penrose pseudo-inverse of the lower-rectangular matrix $\boldsymbol{X}$. Therefore, in view of assumptions *i),ii)* and of standard properties of matrix norm, it is

$$\|\boldsymbol{J}^\dagger(\boldsymbol{q})\| \leq \delta/\beta \triangleq \delta', \ \forall\, \boldsymbol{q} \in Q. \tag{4.17}$$

Alternatively to the jacobian pseudo-inverse algorithm, the jacobian transpose method can be applied when a constant task space desired position is considered, i.e.

$$\boldsymbol{q}_{h+1} = \boldsymbol{q}_h + \gamma T \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}_h)(\boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h)). \tag{4.18}$$

The stability analysis of discrete-time systems similar to those in (4.16) and (4.18) has been tackled in a very few papers (e.g. [27],[28]) based on Lyapunov methods. In Section 4.6.2 an alternative convergence analysis of these algorithms will be given, providing criteria to select the gain γ in relation to the sampling time, as well as an estimation of the region of attraction will be given.

4.6.2 Stability analysis

In the discrete-time version of the CLIK algorithm using the pseudo-inverse of the jacobian (4.16), with a constant $\boldsymbol{x}_d$, the dynamics of the task space error $\boldsymbol{e}_h = \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h)$ is governed by the equation

$$\begin{aligned} \boldsymbol{e}_{h+1} &= \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h + \gamma T \boldsymbol{J}_h^\dagger \boldsymbol{e}_h) \\ &= \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h) - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^\dagger \boldsymbol{e}_h - \boldsymbol{r}_k(\boldsymbol{q}_h) \\ &= (1 - \gamma T)\boldsymbol{e}_h - \boldsymbol{r}_k(\boldsymbol{q}_h), \end{aligned} \tag{4.19}$$

where the notation $\boldsymbol{J}_h = \boldsymbol{J}(\boldsymbol{q}_h)$ has been introduced for brevity, and the expression of $\boldsymbol{k}(\boldsymbol{q})$ in *iii)* has been exploited. The proof of the algorithm convergence and the estimation of the region of attraction depending on γ are addressed by the following theorem. The proof will not make use of any Lyapunov argument as most of the papers dealing with IK of redundant robots (the most recent one is [6]) do. However, when redundant robots are considered, proving that the origin of the task space error space is asymptotically stable requires a Lyapunov function candidate which includes terms depending not only on the task space variables, but also on the configuration variables. The alternative approach followed here completely avoids the use of Lyapunov functions, nevertheless it rigorously proofs the stability of the algorithm. It is based on the following Lemma that is a consequence of the comparison principle for discrete-time systems and is a special case of classical results on recurrence inequalities that can be found in [48].

Lemma 2. *Let b_h be a non-negative sequence satisfying*

$$b_{h+1} \leq \alpha b_h + c, \tag{4.20}$$

where α and c are non-negative real numbers. If $b_0 \leq a_0$, being a_0 the initial condition of the dynamic system

$$a_{h+1} = \alpha a_h + c, \tag{4.21}$$

then

$$b_h \leq a_h, \ \forall\, h \geq 0. \tag{4.22}$$

Proof. The proof is by induction. The claim is true for $h = 0$. Suppose it is true for h, then for $h + 1$ it is

$$b_{h+1} \leq \alpha b_h + c \leq \alpha a_h + c = a_{h+1}. \tag{4.23}$$

□

Theorem 1. *Under the assumptions i)–iii), if the initial task space error $\boldsymbol{e}_0$ and the gain γ are such that*

$$0 < \gamma \leq 1/T \quad \text{and } \|\boldsymbol{e}_0\| < \frac{1}{\gamma T \nu_k \delta'^2} \tag{4.24}$$

or

$$1/T < \gamma < 2/T \text{ and } \|\boldsymbol{e}_0\| < \frac{2 - \gamma T}{\gamma^2 T^2 \nu_k \delta'^2}, \tag{4.25}$$

then the CLIK algorithm in (4.16) ensures exponential convergence of the task space error dynamics and the configuration variables $\boldsymbol{q}_h$ are bounded and converge to a constant value, i.e.

$$\exists\, \alpha \in (0,1),\, \phi > 0 : \quad \|\boldsymbol{e}_h\| \leq \phi \alpha^h, \ \forall h \geq 0 \tag{4.26}$$

$$\exists\, \rho > 0 : \quad \|\boldsymbol{q}_h\| \leq \rho, \ \forall h \geq 0 \tag{4.27}$$

$$\lim_{h \to \infty} \|\boldsymbol{q}_{h+1} - \boldsymbol{q}_h\| = 0. \tag{4.28}$$

Proof. From (4.19) the following inequalities are obtained

$$\begin{aligned} \|\boldsymbol{e}_{h+1}\| &\leq |1 - \gamma T| \|\boldsymbol{e}_h\| + \|\boldsymbol{r}_k(\boldsymbol{q}_h)\| \\ &\leq |1 - \gamma T| \|\boldsymbol{e}_h\| + \nu_k \|\gamma T \boldsymbol{J}_h^\dagger \boldsymbol{e}_h\|^2 \\ &\leq (|1 - \gamma T| + \gamma^2 T^2 \nu_k \delta'^2 \|\boldsymbol{e}_h\|) \|\boldsymbol{e}_h\|, \end{aligned} \tag{4.29}$$

where the bounds in assumption *iii)* and in Eq. (4.17) have been exploited together with standard norm properties. Now, assume that the task error norm is bounded, i.e.

$$\|\boldsymbol{e}_h\| \leq \phi, \ \forall\, h \geq 0. \tag{4.30}$$

In a moment it will be shown that such a condition is guaranteed by the sole hypothesis that the initial condition satisfies one of the two conditions (4.24) or (4.25). Equation (4.29) becomes

$$\|\boldsymbol{e}_{h+1}\| \leq (|1-\gamma T| + \gamma^2 T^2 \nu_k \delta'^2 \phi)\|\boldsymbol{e}_h\| \triangleq \alpha \|\boldsymbol{e}_h\|, \ \forall\, h \geq 0. \tag{4.31}$$

Firstly, assume that the gain and the initial task space error satisfy (4.24), by choosing $\phi = \|\boldsymbol{e}_0\|$ it is

$$\phi < \frac{1}{\gamma T \nu_k \delta'^2} \ \Rightarrow \ \alpha < 1, \tag{4.32}$$

hence the following scalar linear system

$$\tilde{e}_{h+1} = \alpha \tilde{e}_h \tag{4.33}$$

is asymptotically stable and its response with initial condition $\tilde{e}_0 = \phi$ is

$$\tilde{e}_h = \phi \alpha^h, \ h \geq 0. \tag{4.34}$$

Therefore, in view of (4.31) and recalling that $\|\boldsymbol{e}_0\| = \tilde{e}_0 = \phi$, from Lemma 2 it results

$$\|\boldsymbol{e}_h\| \leq \phi \alpha^h, \ \forall\, h \geq 0, \tag{4.35}$$

which proves (4.26) and ensures also that (4.30) is verified. To conclude the proof, the boundedness of $\boldsymbol{q}_h$ can be easily shown by considering Eq. (4.16) and the following chain of inequalities

$$\|\boldsymbol{q}_{h+1}\| \leq \|\boldsymbol{q}_h\| + \|\gamma T \boldsymbol{J}_h^{\dagger} \boldsymbol{e}_h\| \leq \|\boldsymbol{q}_h\| + \gamma T \delta' \phi \alpha^h, \tag{4.36}$$

where Eqs. (4.17) and (4.35) have been exploited. Now, consider the scalar linear system

$$\tilde{q}_{h+1} = \tilde{q}_h + \gamma T \delta' \phi \alpha^h, \tag{4.37}$$

whose response with initial condition $\tilde{q}_0 = \|\boldsymbol{q}_i\|$ is

$$\tilde{q}_h = \tilde{q}_0 + \frac{\gamma T \delta' \phi}{1-\alpha}(1-\alpha^h). \tag{4.38}$$

Again, in view of (4.36),(4.38) and by applying Lemma 2, it results

$$\|\boldsymbol{q}_h\| \leq \tilde{q}_h \leq \tilde{q}_0 + \frac{\gamma T \delta' \phi}{1-\alpha} = \rho, \ \forall\, h \geq 0, \tag{4.39}$$

which proves the claim (4.27). To conclude the proof, observe that from (4.16) it follows that

$$\|\boldsymbol{q}_{h+1} - \boldsymbol{q}_h\| \leq \gamma T \|\boldsymbol{J}_h^{-1}\| \|\boldsymbol{e}_h\| \leq \gamma T \delta' \|\boldsymbol{e}_h\|, \tag{4.40}$$

and thus, in view of (4.35), the last claim (4.28) immediately follows. If the gain γ and the initial task error $\boldsymbol{e}_0$ satisfy assumption (4.25), the proof is perfectly analogous. □

The Theorem above clearly shows that the gain of the CLIK algorithm has to be chosen in relation to the sampling time and, as expected, the lower is the sampling time, the larger can be selected the gain. Moreover, the bound on the initial task space error allows to estimate the region of attraction of the origin of the task space error space, which can be enlarged only by reducing such gain or by staying far from singularities. Finally, the upper limit to the gain $2/T$ is certainly given by a sufficient condition, although it does not appear too restrictive as confirmed by the simulations presented in Section 4.6.3. Furthermore, the theorem ensures that no inner motion can occur even if no redundancy resolution scheme is adopted (see claim (4.28) of Theorem 1). Finally, the exponential convergence condition (4.26) implies the local exponential asymptotical stability of the origin of the task space error space.

Now, the convergence of the discrete-time IK algorithm, which makes use of the jacobian transpose, i.e. Eq. (4.18), will be proved. The analysis will be carried out by resorting to the same methodology used in the previous theorem and to the following Lemma.

Lemma 3. *Let $\boldsymbol{H}$ be a full rank $m \times n$ matrix, with $m \leq n$, then*

$$\|\boldsymbol{I} - \eta \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\| = 1 - \eta \underline{\sigma}(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}), \quad \forall \eta \in [0, 1/\|\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\|). \tag{4.41}$$

Proof. Consider the singular value decomposition $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, the assumption that $\boldsymbol{H}$ is full rank implies that $\boldsymbol{\Sigma} = \mathrm{diag}\{\sigma_1, \ldots, \sigma_m\}$ is such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m > 0$ and of course it is $\boldsymbol{\Sigma}^2 = \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_m^2\}$, where $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \sigma_m^2 > 0$ are the singular values of $\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$, whose norm is therefore σ_1^2. Furthermore, it is

$$\begin{aligned} &\boldsymbol{I} - \eta \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{I} - \eta \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{I} - \eta \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^{\mathrm{T}} \\ &= \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} - \eta \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}(\boldsymbol{I} - \eta\boldsymbol{\Sigma}^2)\boldsymbol{U}^{\mathrm{T}}, \end{aligned} \tag{4.42}$$

where the matrix $\boldsymbol{I} - \eta\boldsymbol{\Sigma}^2 = \mathrm{diag}\{1 - \eta\sigma_1^2, \ldots, 1 - \eta\sigma_m^2\}$ is such that $1 - \eta\sigma_m^2 \geq 1 - \eta\sigma_{m-1}^2 \geq \ldots \geq 1 - \eta\sigma_1^2 > 0 \; \forall \eta \in [0, 1/\sigma_1^2)$, thus the decomposition (4.42), except for a simple reordering, is the singular value decomposition of $\boldsymbol{I} - \eta\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$ whose norm is therefore $1 - \eta\sigma_m^2 = 1 - \eta\underline{\sigma}(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})$. □

The task space error dynamics of the algorithm (4.18) is governed by the equation

$$\begin{aligned} \boldsymbol{e}_{h+1} &= \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h + \gamma T \boldsymbol{J}_h^{\mathrm{T}} \boldsymbol{e}_h) \\ &= \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}_h) - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}} \boldsymbol{e}_h - \boldsymbol{r}_k(\boldsymbol{q}_h) \\ &= (\boldsymbol{I} - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}})\boldsymbol{e}_h - \boldsymbol{r}_k(\boldsymbol{q}_h), \end{aligned} \tag{4.43}$$

where the expression of $\boldsymbol{k}(\boldsymbol{q})$ in *iii)* has been exploited again. The following theorem carries out the convergence analysis of this algorithm and provides a bound to the gain γ.

Theorem 2. *Under the assumptions i),ii),iii), if the initial task space error* $\boldsymbol{e}_0$ *and the gain* γ *are such that*

$$0 < \gamma < \frac{1}{T\delta^2} \quad \textit{and} \quad \|\boldsymbol{e}_0\| < \frac{\beta}{\gamma T \nu_k \delta^2}, \tag{4.44}$$

then the CLIK algorithm in (4.18) ensures exponential convergence of the task space error dynamics and the configuration variables $\boldsymbol{q}_h$ *are bounded and converge to a constant value, i.e.*

$$\exists\, \alpha \in (0,1),\, \phi > 0 : \qquad \|\boldsymbol{e}_h\| \leq \phi \alpha^h,\ \forall h \geq 0 \tag{4.45}$$

$$\exists\, \rho > 0 : \qquad \|\boldsymbol{q}_h\| \leq \rho,\ \forall h \geq 0 \tag{4.46}$$

$$\lim_{h\to\infty} \|\boldsymbol{q}_{h+1} - \boldsymbol{q}_h\| = 0. \tag{4.47}$$

Proof. From (4.43) and, owing to assumption (4.44), by applying Lemma 3 to compute the norm of the matrix $\boldsymbol{I} - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}}$, the following inequalities are obtained

$$\begin{aligned}
\|\boldsymbol{e}_{h+1}\| &\leq \|\boldsymbol{I} - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}}\| \|\boldsymbol{e}_h\| + \|\boldsymbol{r}_k(\boldsymbol{q}_h)\| \\
&\leq (\|\boldsymbol{I} - \gamma T \boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}}\| + \gamma^2 T^2 \nu_k \delta^2 \|\boldsymbol{e}_h\|) \|\boldsymbol{e}_h\| = \\
&= (1 - \gamma T \underline{\sigma}(\boldsymbol{J}_h \boldsymbol{J}_h^{\mathrm{T}}) + \gamma^2 T^2 \nu_k \delta^2 \|\boldsymbol{e}_h\|) \|\boldsymbol{e}_h\| \leq \\
&\leq (1 - \gamma T \beta + \gamma^2 T^2 \nu_k \delta^2 \|\boldsymbol{e}_h\|) \|\boldsymbol{e}_h\|,
\end{aligned} \tag{4.48}$$

where the bounds in assumptions *i),ii),iii)* have been exploited together with standard norm properties. Now assume that the task error norm is bounded, i.e.

$$\|\boldsymbol{e}_h\| \leq \phi,\ \forall\, h \geq 0. \tag{4.49}$$

In a moment it will be shown that such a condition is guaranteed by the sole hypothesis that the initial condition satisfies condition (4.44). Equation (4.48) becomes

$$\|\boldsymbol{e}_{h+1}\| \leq (1 - \gamma T \beta + \gamma^2 T^2 \nu_k \delta'^2 \phi) \|\boldsymbol{e}_h\| \triangleq \alpha \|\boldsymbol{e}_h\|,\ \forall\, h \geq 0. \tag{4.50}$$

Since, by assumption, the gain and the initial task space error satisfy (4.44), by choosing $\phi = \|\boldsymbol{e}_0\|$ it is

$$\phi < \frac{\beta}{\gamma T \nu_k \delta^2} \ \Rightarrow\ \alpha < 1. \tag{4.51}$$

The rest of the proof is perfectly analogous to the one of Theorem 1 and thus it is omitted for brevity. □

Note that, differently from the stability analysis performed in [27], there is no necessity to include any additional matrix to avoid inner motions. In fact, no inner motion can occur since the configuration variables are shown to reach a steady-state (see claim (4.47) in Theorem 2). Moreover, also in this case the limit on the gain γ is related to the sampling time and to the robot kinematics. In particular, the closer is any kinematic singularity to the configuration space Q, the lower is the bound β and thus the lower can be selected the gain γ if the same amplitude of the region of attraction is desired. On the other hand, the gain can be increased if a smaller region of attraction can be tolerated.

4.6.3 Simulations

Two simulation cases have been presented to show the practical utility of the results obtained.

The first case study is the classical inverse kinematics problem for a three-link planar manipulator, and the simulations are intended to show how the sufficient conditions of Theorem 1 are not conservative at all. The simulations were carried out in MATLAB/Simulink using the Embedded MATLAB toolbox. solutions [67], thus the

Assuming 1 m long links, the direct kinematics function $\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$ is

$$x_1 = \cos(q_1) + \cos(q_1 + q_2) + \cos(q_1 + q_2 + q_3) \tag{4.52}$$

$$x_2 = \sin(q_1) + \sin(q_1 + q_2) + \sin(q_1 + q_2 + q_3) \tag{4.53}$$

where $\boldsymbol{q} = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix}^{\mathrm{T}}$ is the vector of joint variables and $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^{\mathrm{T}}$ is the vector of task space variables, i.e. the Cartesian position of the robot end effector (x_1, x_2). The robot jacobian, not reported for brevity, is always full rank if, e.g. the joint variables q_2 and q_3 are constrained to assume values in the interval $[-3/4\pi, -\pi/4]$. Within this joint space, it is easy to numerically evaluate the constants $\delta = 3.09$ and $\beta = 9.58$ in assumptions *i), ii)*, respectively. It is also evident that both the direct kinematics function and all the columns of the jacobian have norm-bounded Hessian matrices. Therefore, assumption *iii)* is verified and the constant $\nu_k = 3.09$ is easily estimated recalling its definition in Lemma 1, i.e. $\nu_k = \sqrt{2}/2 \max\{\nu_1, \nu_2\}$, being ν_1 and ν_2 the upperbounds of the Hessian matrices norm of the function $\boldsymbol{k}(\boldsymbol{q})$. These constants are useful to estimate the regions of attraction of the two algorithms, i.e. the maximum allowed initial task space error norm, according to Eqs. (4.24),(4.25) and (4.44), respectively.

The desired constant position in the task space is $\boldsymbol{x}_d = \begin{pmatrix} -0.3 & 0.4 \end{pmatrix}^{\mathrm{T}}$. The considered sampling time is $T = 1$ ms, while the initial robot configuration is

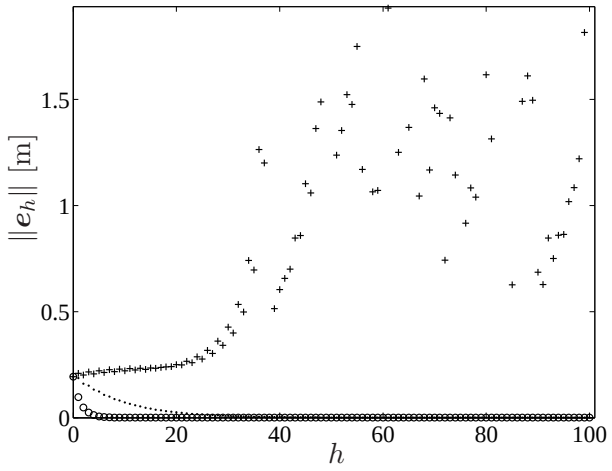


Figure 4.14: CLIK algorithm based on the jacobian pseudo-inverse: norm of the task space error with $\gamma = 0.5/T$ (circles), $\gamma = 1.9/T$ (dots) and $\gamma = 2.01/T$ (crosses).

$\boldsymbol{q}_i = (-\pi \quad -\pi/2 \quad -2\pi/3)^{\mathrm{T}}$. The robot is used in the following simulations to show that the sufficient condition on the algorithm gain γ of Theorems 1 is not too conservative. The algorithm based on the jacobian pseudo-inverse (Eq. (4.16)) has been tested with three values of the gain, i.e. $\gamma = 0.5/T$, $\gamma = 1.9/T$ and $\gamma = 2.01/T$, the last one just a little bit over the limit in (4.25). The resulting task space error norm is reported in Fig. 4.16 for all gain values. It is evident how the algorithm converges only with the gain values less that $2/T$. The results obtained by using the inverse kinematics algorithm with the jacobian transpose are reported in Fig. 4.18, which evidently shows that the sufficient condition found in Theorem 2 are more conservative than those found in Theorem 1, since with a gain γ equal to $1/(\beta T)$ the algorithm still converges, whereas the minimum value of the gain γ leading to an unstable behaviour of the algorithm is $20/(\beta T)$.

The second case study is the IK problem for the eleven-link humanoid manipulator described in Section 2.9.9 of [67]. It is constituted by a four-link torso and a seven-link arm. The simulations are intended to show how the sufficient conditions of Theorem 1 are not restrictive at all. The simulations were carried out in MATLAB/Simulink using the Robotics Toolbox [22]. The forward kinematics of the robot has been obtained by solving the exercise 2.14 of [67]. In order to perform the simulation, a subset Q of the joint space has been considered, in which the jacobian matrix of the manipulator is full-rank. Within this joint space, it is easy to evaluate, through a standard nonlinear optimization algorithm, the constants $\delta = 4.19$ and $\beta = 0.076$, in

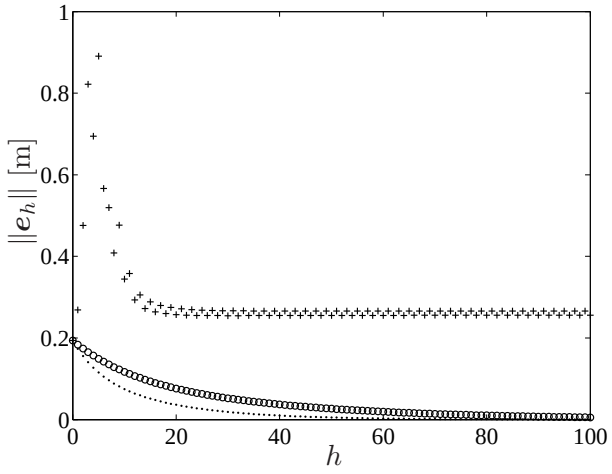


Figure 4.15: CLIK algorithm based on the jacobian transpose: norm of the task space error with $\gamma = 0.5/(\beta T)$ (circles), $\gamma = 1/(\beta T)$ (dots) and $\gamma = 20/(\beta T)$ (crosses).

assumptions *i),ii)*, respectively. To verify that the direct kinematics function have norm-bounded Hessian matrices, namely that assumption *iii)* holds, the constant $\nu_k = 9.17$ is easily estimated recalling its definition in Lemma 1, i.e. $\nu_k = \sqrt{2}/2 \max\{\nu_i\}$, being ν_i the upper bound of the i-th Hessian matrix norm of the function $\boldsymbol{k}(\boldsymbol{q})$. These constants are useful to estimate the regions of attraction of the two algorithms, i.e. the maximum allowed initial task space error norm, according to Eqs. (4.24),(4.25),(4.44).

The desired constant position in the task space is

$$\boldsymbol{x}_d = \begin{pmatrix} -0.071 & -0.639 & 0.507 & 1.48 & 0.202 & -2.22 \end{pmatrix}^{\mathrm{T}}$$

, in which the orientation is expressed by using the ZYZ Euler angles. The considered sampling time is $T = 1\,\mathrm{ms}$, while the initial robot configuration is $\boldsymbol{q}_i = \begin{pmatrix} \pi/2 & \pi/3 & \pi/3 & 0 & \pi/4 & \pi/4 & -\pi/4 & 2/3\pi & 3/4\pi & 3/4\pi & 0 \end{pmatrix}^{\mathrm{T}}$. The robot is used in the following simulations to show that the bounds on the algorithm gain γ of Theorems 1 are not too restrictive. The algorithm based on the jacobian pseudo-inverse (Eq. (4.16)) has been tested with three values of the gain, i.e. $\gamma = 0.5/T$, $\gamma = 1.9/T$ and $\gamma = 2.01/T$, the last one just a little bit beyond the bound in (4.25). In this simulation, the algorithm includes the redundancy resolution, performed by resorting to the classical technique of the projector into the jacobian null space, i.e.

$$\boldsymbol{q}_{h+1} = \boldsymbol{q}_h + \gamma T \boldsymbol{J}_h^{\dagger} \boldsymbol{e}_h + T(\boldsymbol{I} - \boldsymbol{J}_h^{\dagger} \boldsymbol{J}_h) \dot{\boldsymbol{q}}_{0_h}, \tag{4.54}$$

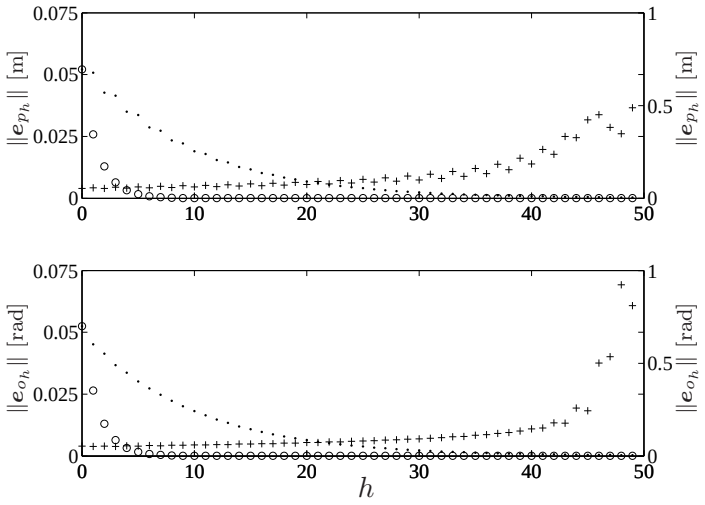


Figure 4.16: CLIK algorithm based on the jacobian pseudo-inverse: norm of the task space position and orientation error with $\gamma = 0.5/T$ (circles, left y-axis), $\gamma = 1.9/T$ (dots, left y-axis) and $\gamma = 2.01/T$ (crosses, right y-axis).

where $\dot{\boldsymbol{q}}_{0_h} = -\nabla(w(q_h))^{\mathrm{T}}$ and $w(q)$ is a cost function to minimize [67]. The distance in the joint space from the center of joint ranges has been chosen as cost function. The resulting task space error norm is reported in Fig. 4.16 for all gain values, showing how the algorithm converges only with the gain values less than $2/T$. Figure 4.17 reports the time history of the cost function with and without the redundancy resolution, and, in the former case, better values are obtained. The results obtained by using the IK algorithm with the jacobian transpose are reported in Fig. 4.18, which shows that the gain bound found in Theorem 2 is more restrictive than those found in Theorem 1, since with a gain γ equal to $1/(\delta^2 T)$ the algorithm still converges, whereas the minimum value of the gain γ leading to an unstable behavior of the algorithm is $20/(\delta^2 T)$.

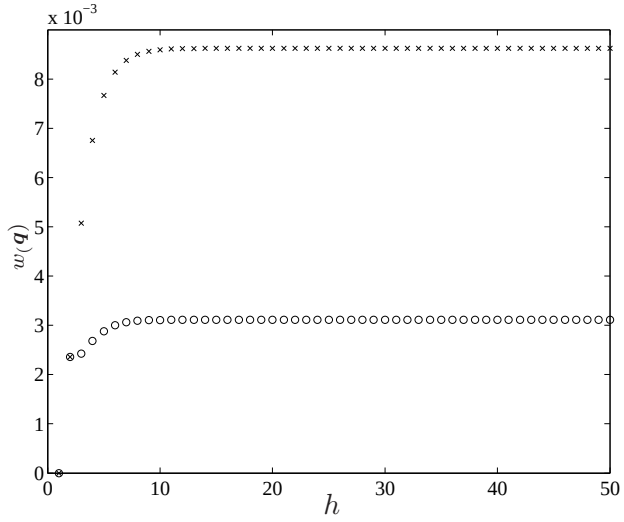


Figure 4.17: Cost function obtained from jacobian pseudo-inverse algorithm with redundancy resolution (circles) and without redundancy resolution (crosses).

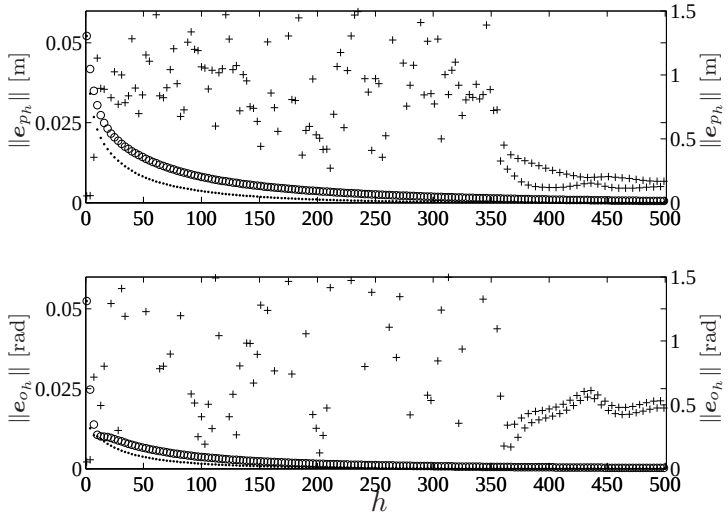


Figure 4.18: CLIK algorithm based on the jacobian transpose: norm of the task space position and orientation error with $\gamma = 0.5/(\delta^2 T)$ (circles, left y-axis), $\gamma = 1/(\delta^2 T)$ (dots, left y-axis) and $\gamma = 20/(\delta^2 T)$ (crosses, right y-axis).

Chapter 5

Applications in robot calibration

Exploiting the results obtained for the hand calibration step, described in Chapter 3, a procedure for estimating the kinematic parameters of any industrial serial manipulator has been developed. The novelty of the procedure consists in the adoption of a motion capture system, that allows a 3D position measurement distributed along the entire kinematic chain not only of its end effector. This enables the estimation of the kinematic parameters in a closed form without the need to resort to any linearisation of the error function. The robustness of the estimation algorithm against measurement noise is guaranteed by the adoption of multivariate statistical methods like the Principal Component Analysis (PCA). The procedure is intended for robot manufactures, who can adopt it to select the estimated kinematic parameters, to be used in the control units of the robot for the computation of both direct and inverse kinematics, for improving the absolute positioning accuracy of the end effector. Both simulation and experimental results obtained in the calibration of an actual industrial 6DOF robot are reported, which confirm the effectiveness of the proposed approach. The off-line programming method of industrial robotic work cells is rapidly catching on, enabled by modern CAD systems. These software packages allow the user not only to draw the layout of the work cell but also to carry out the simulation of the motion of robot manipulators and machine tools performing the manufacturing tasks. This feature represents a real breakthrough in the entire manufacturing process since it has an impact in reducing the overall life cycle cost of the product being manufactured. In fact, the commissioning cost of the work cell is reduced as well as the switch-off time of the robot for re-programming is drastically reduced, since the simulated robot program can be directly downloaded into the robot control system. One of the main problems for making the off-

line programming method effective in practice is the absolute accuracy of the robot motion within the work cell. The major source of inaccuracy is the difference between the kinematic parameters used in the robot control unit and the actual ones. The former are usually based on the robot design specifications, while the real parameters are affected by manufacturing tolerances, mounting errors during robot link assembly. Differences between nominal and real kinematic models also rise from nongeometrical errors e.g., link-and-joint flexibility, backlash, gear wear. Robot kinematic calibration consists of identifying a more accurate geometrical relationship between the joint position sensors readings and the actual position of the end-effector; then the robot positioning software is changed according to this relationship identified. Kinematic calibration involves four steps: modelling, measurement, identification, and correction [43]. The contribution of the present method mainly resides in the second and third steps. The measurement system proposed here can be any commercial optical motion capture system (mocap). Mocap systems are constituted by a set of calibrated cameras (normally infrared) and a workstation able to reconstruct the 3D position of markers with a typical accuracy of the order of 0.1 mm. The cost of the system is mainly affected by the resolution, speed and number of the cameras. For the application at hand, neither the resolution nor the speed are relevant parameters, since large markers can be easily placed on an industrial robot and thus a small number (4 or 5 are usually enough) low resolution cameras can be used to capture the typical workspace of an industrial robot. The speed of acquisition is certainly much lower than the one required by the most common application of these systems (recording action of human actors in movie industry for subsequent computer animation of digital characters) since the movements required to the robot during the calibration procedure can be very slow. Therefore, with a limited number of low resolution and low speed cameras, the cost of the measurement system can be even lower than the cost of the machine to be calibrated. On the basis of the proposed measurement system, the proposed method can be classified as an open-loop method, thus does not require any special machinery and addition calibration tool or fixture, again in the interest of the ease and economy of application, in contrast to closed-loop methods which are generally more expensive and time consuming.

Besides the limited cost needed by the metrology, the additional advantage of the motion capture system combined with the specific calibration technique proposed here is the possibility of measuring not only the position of the robot end effector but of many other points selected on the links constituting the serial kinematic chain. This feature suggested the procedure itself, which basically consists in capturing the motion of a number of points along

the robot structure that allow the estimation of the rotational or translational (depending on the joint type: revolute or prismatic) axes of the robot joints; then simple geometric relationships can be used to derive the kinematic parameters. Similar approaches are used in biomechanics to estimate the parameters of kinematic models of the human skeleton, e.g. [38, 79, 12, 35]. For robot kinematic calibration, the same idea was firstly proposed in [9], where a single point fixed on the robot end effector was tracked. Later on, in [70] a similar set up was applied to calibrate a PUMA 560 robot by means of acoustic sensors mounted on each robot link. A very similar method is also known as Circle Point Analysis (CPA), applied using a theodolite system in [69]. However, differently from the approach pursued here, in those early works the plane of rotation and the center of rotation were estimated through ordinary least square methods. In the present work, the use of the Principal Component Analysis as a robust multivariate statistic algorithm is proposed, that is able to provide accurate estimates of spatial directions of minimum and maximum variance of data, with a low sensitivity to measurement noise [44], which is particularly relevant in kinematic calibration [42]. In general, PCA finds a p-dimensional linear manifold minimising a scale of the orthogonal distances of the m-dimensional data points to the manifold. Therefore, with respect to ordinary least square fitting, PCA does not assume that noise affects only observed data but it takes into account also the case that the regressor is affected by noise, hence providing better robustness to measurement noise.

Several advantages characterise the geometric method in comparison with classical iterative methods. The algorithm can be used even when large errors affect kinematic parameters, since no prior knowledge of the design parameters is needed in the initialisation phase of the algorithm, as usual in the methods which only adjust these parameters to the extent that the measurements provide sufficient evidence, both for serial kinematic chains [77] and for methods suitable for parallel robots [62, 14]. Moreover, the identified joint axes maybe expressed with respect to any coordinate system, since only the relative position of two consecutive lines is relevant. Also, there is no need to select poses to ensure full column rank of the error jacobian, as usually happens for all the methods that lead to a solution based on optimisation algorithms [81]. Finally, the adoption of a motion capture system allows to calibrate also the absolute position of the robot, which is a key requirement for an off-line programming system.

The rest of this chapter is organised as follows. Section 5.1 summarises the kinematic model of the robot to be calibrated. It exploits the DH convention due to its large diffusion in industrial robots, and suitably modified in case of near parallel consecutive joint axes. Section 5.3 describe the details

of the proposed calibration procedure both in terms of measurement process and of data elaboration. The motion capture system used in the experiments is detailed in Section 5.2. The features of the algorithm in terms of attainable performance in absence and presence of measurement noise are firstly discussed in simulation in Section 5.4, then they are actually proved by the experiments described in Section 5.5. The calibration of an actual industrial robot is carried out and the results show that using the calibrated parameters allows to increase the absolute accuracy of the robot.

5.1 Kinematic model

Given a generic serial chain manipulator with n joints, assuming to select n frames, each attached to a link of the robot according to the Denavit-Hartenberg (DH) convention [30], the pose of the frame attached to the last link of the robot can be described by the homogenous transformation matrix

$$\boldsymbol{T}_n^0(\boldsymbol{q}) = \boldsymbol{T}_1^0(q_1)\boldsymbol{T}_2^1(q_2)\cdots\boldsymbol{T}_n^{n-1}(q_n), \tag{5.1}$$

where $\boldsymbol{T}_i^{i-1}(q_i)$ is the transformation matrix expressing the pose of frame i with respect to frame $i-1$, q_i is the joint variable of joint i and $\boldsymbol{q}$ is the vector of the n joint variables. The analytic expression of this matrix as a function of the DH parameters is [68]

$$\boldsymbol{T}_i^{i-1}(q_i) = \begin{pmatrix} \cos\vartheta_i & -\sin\vartheta_i\cos\alpha_i & \sin\vartheta_i\sin\alpha_i & a_i\cos\vartheta_i \\ \sin\vartheta_i & \cos\vartheta_i\cos\alpha_i & -\cos\vartheta_i\sin\alpha_i & a_i\sin\vartheta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5.2}$$

where ϑ_i, α_i, a_i and d_i are the DH parameters of link i and q_i is the i-th joint variable, i.e.

$$q_i = \begin{cases} \vartheta_i & \text{revolute joint} \\ d_i & \text{prismatic joint} \end{cases}.$$

Hereafter, it is assumed that only the kinematic parameters, except joint variables, are subject to calibration, since for the determination of joint offsets specific procedures exist [17], or they are usually foreseen by the standard maintenance procedure of the robot. In case two consecutive joint axes are near parallel, suppose $j-1$ and j, the modified DH convention proposed in [40] is adopted and the direct kinematic equation 5.1 is modified by introduction an additional rotation of an angle β_j about y axis of the current frame. In detail, for a revolute joint the homogenous transformation reduces to

$$\hat{\boldsymbol{T}}_j^{j-1}(q_j) = \begin{pmatrix} \cos\vartheta_j\cos\beta_j - \sin\alpha_j\sin\vartheta_j\sin\beta_j & -\sin\vartheta_j\cos\alpha_j \\ \sin\vartheta_j\cos\beta_j + \sin\alpha_j\cos\vartheta_j\sin\beta_j & \cos\vartheta_j\cos\alpha_j \\ -\cos\alpha_j\sin\beta_j & \sin\alpha_j \\ 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} \cos\vartheta_j\sin\beta_j + \sin\vartheta_j\sin\alpha_j\cos\beta_j & a_j\cos\vartheta_j \\ \sin\vartheta_j\sin\beta_j - \cos\vartheta_j\sin\alpha_j\cos\beta_j & a_j\sin\vartheta_j \\ \cos\alpha_j\cos\beta_j & 0 \\ 0 & 1 \end{pmatrix} \tag{5.3}$$

Note that β_j does not increase the number of parameter to be actually identified, since if joint j is revolute, the parameter d_j is no longer needed. If joint j is prismatic, then d_j is the joint variable and the homogenous transformation reduces to

$$\hat{\boldsymbol{T}}_j^{j-1}(q_j) = \begin{pmatrix} \cos\beta_j & 0 & \sin\beta_j & 0 \\ \sin\alpha_j\sin\beta_j & \cos\alpha_j & -\sin\alpha_j\cos\beta_j & 0 \\ -\cos\alpha_j\sin\beta_j & \sin\alpha_j & \cos\alpha_j\cos\beta_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.4}$$

and, again, the number of parameters to be identified is not increased. The choice made here of the DH convention for expressing the end-effector pose as a function of the joint variables and the kinematic parameters has been done because it is universally used in industrial robots and the proposed calibration procedure is specifically intended for adoption by the robot manufacturers to enhance the accuracy of their products. Alternative kinematic modelling procedures exist, like those based on the Product Of Exponentials (POE) formula [41, 18]. Such methods have been proposed to ensure a smooth mapping between the kinematic parameters and the identification error, thus making identification algorithms robust and singularity. Nevertheless, the calibration procedures proposed based on this modelling methodology still need local linearisation and algorithm iterations to get the unknown parameters, which, also, have no clear geometrical meaning. Furthermore, even in absence of measurement noise, the algorithms does not guarantee to obtain the true values of the kinematic parameters. In fact, as clearly stated in [41], "the iterative algorithm for parameter identification is convergent when the deviations of the actual kinematic parameters from the nominal values are small, and the pose measurements are reasonably chosen to make the identification-Jacobian-matrix column full rank". These are the typical limitations of all the algorithms based on local linearisation of the identification error. On the contrary, this method adopts a procedure leading to the identification

of geometrically meaningful kinematic parameters in closed form. Also, as it will be detailed in the next section, the presence of measurement noise is counteracted by the purposeful adoption of statistical methods.

For the reader's convenience, it is now recalled the geometric meaning of the DH kinematic parameters summarised in the well-known DH table Tab. 5.1.

Link	a_i	α_i	d_i	ϑ_i
1	a_1	α_1	d_1	ϑ_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	a_n	α_n	d_n	ϑ_n

Table 5.1: DH table

In particular

- a_i is the distance between the axes of joints i and $i+1$, it is therefore computed along the common normal of these two axes;
- d_i is the quote with respect to the frame fixed to link $i-1$ of the point of intersection of the common normal cited above and the axis of joint i;
- α_i is the angle between the axes of joints i and $i+1$ about the x axis of the frame fixed to link i;
- ϑ_i is the angle between the x axes of the frames fixed to links $i-1$ and i

In a similar way the Hayati parameter β_i is defined in case of near parallel consecutive joint axes. It is evident that once the directions in the 3D space of all the joint axes and the position of one point of each of these axes are known, then the kinematic parameters can be computed quite easily as it will be detailed in Section 5.3.4.

5.2 Measurement system

The main idea is to estimate the directions of the joint axes by moving the joints one by one. This implies that, if the joint is revolute, each point of the following robot links moves on a circumference and all these circumferences lies on parallel planes and the centres lay on the joint axis. If the joint is prismatic, then each point of the following robot links moves along a straight

Figure 5.1: Dynamic (left) and static (right) calibrators of a VICON motion capture system.

line and all these straight lines are parallel and identify the direction of the joint axis. Therefore, by measuring the path of at least one of such points, the joint axes can be easily estimated for both type of joints. It is clear that by measuring the path of a larger number of points, a higher robustness against measurement noise or outliers is expected. Another key point which makes the procedure quite easy to execute is that the exact location on the link of each point to be tracked is not relevant, any point can be used; some suggestions on how to make the selection will be given in the following. This simple idea suggested the use of a marker-based motion capture system, which is a natural candidate to track the motion of selected points in the 3D space, i.e. the points where the markers are placed.

In fact, an optical motion capture system is composed by two or more cameras (in the experimental calibration performed in Section 5.5 only four cameras were used), a workstation connected to a host PC and a set of reflective markers. Usually, such markers are covered by a material with a high reflection index in the wavelength range of the cameras, typically the infrared range. Also, to minimise the effect of lighting conditions each camera is equipped with a light strobe synchronised with the camera shutter. These systems are usually very easy to use since are designed also for non expert users. The steps to perform a capture session are the following:

- **placement of markers and cameras**: this step is the most time consuming since the objective is to maximise the field of view of all the cameras as well as the number of markers visible by at least two cameras (the minimum number needed to reconstruct the 3D position of a marker);

- **camera calibration**: firstly a static calibration is needed to establish the measurement reference frame (see the static calibrator in Fig. 5.1), then a dynamic calibration is needed to estimate both the intrinsic and the extrinsic parameters of the cameras. The classical "wand" of a VICON motion capture system can be seen in Fig. 5.1; the user has to move the wand all over the capture volume so as to make it visible to all the cameras to calibrate. The entire calibration process

for a $1\,m^3$ capture volume usually takes only a couple of minutes. The result of the calibration phase usually provides residues of the markers reconstructed position of the order of 0.1 mm, which is an estimate of the accuracy of the measurement system.

- **data acquisition**: this is the actual motion capture session which can be started and terminated automatically based on an external trigger or manually by the user;
- **marker labelling**: this step consists in assigning a label to each marker so that each marker can be tracked uniquely; then a motion hierarchy of all the markers can be also defined by specifying the so-called "skeletal model", which is not needed for the kinematic calibration procedure proposed in this work, because all the markers move at the same time since the robot joints are moved one at a time;
- **data recording**: the data can be saved in various format, but the most common is the `c3d` standard [1], where the user has an easy access to the 3D coordinates of all the labelled markers .

5.3 Kinematic calibration procedure

As recalled in the previous section, the procedure firstly needs to estimate the directions of the joint axis, and this is accomplished by resorting to the PCA, both for revolute and prismatic joints. Then, the the axis have to be located in the 3D space and this problem is solved by estimating the center of circumference for revolute joints, while it is trivial for prismatic joints as it will be clear later. The final step is to compute the kinematic parameters simply by applying their geometric definition given in Section 5.1.

5.3.1 Data acquisition

Once the robot configuration is selected by following the suggestions given in the previous section, the robot is programmed to move one joint at a time with displacements as large as possible compatibly with the capture volume observable by the cameras. The joint speed should be low enough to let the motion capture system collect a number of frames, which, as a rule of thumb, should range between 1000 and 3000. Modern mocap systems are fast enough to let the robot move at full speed, but of course this is not advisable to prevent exciting link and joint flexibility causing unwanted vibrations. As said before, on each link of the robot one or more markers can

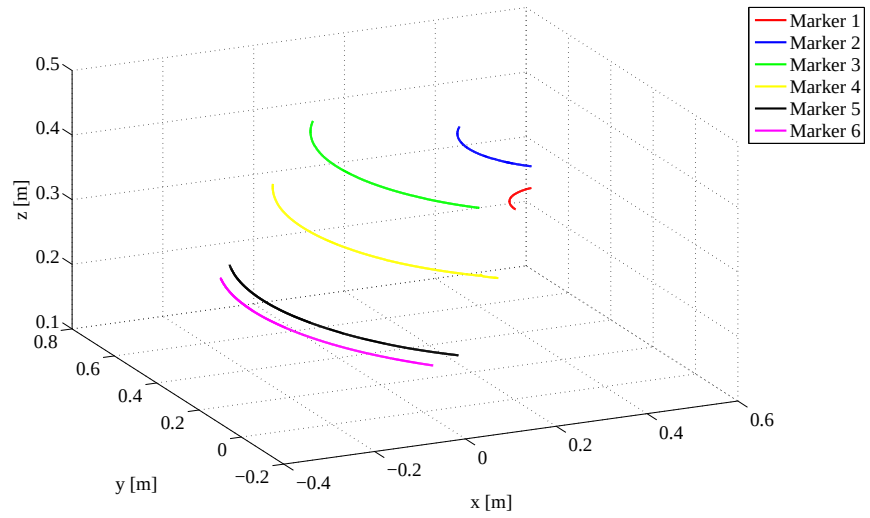


Figure 5.2: Typical marker trajectories captured during the motion of one revolute joint.

be attached, thus if M_i is the number of markers attached to the link i and all the following links, when the joint i is moved there are M_i markers that move on a circumference. Assuming to collect the same number N of frames for all the n joints, once all the frames when the 3D position of at least one marker has not been reconstructed (due to an occlusion) have been removed from the capture trial[1], then the data set can be organised as follows. Let $\widetilde{\boldsymbol{C}}_{ij}$ be the $N \times 3$ matrix of the reconstructed positions expressed in the measurement frame of marker j when joint i is actuated, i.e.

$$\widetilde{\boldsymbol{C}}_{ij} = \begin{pmatrix} x_{ij1} & y_{ij1} & z_{ij1} \\ \vdots & \vdots & \vdots \\ x_{ijN} & y_{ijN} & z_{ijN} \end{pmatrix}, \quad j = 1, \ldots, M_i, \quad i = 1, \ldots, n. \tag{5.5}$$

Typical recorded paths are reported in Fig. 5.2 for a revolute joint.

5.3.2 Identification of joint axes directions

In this step the direction of each joint axis is estimated by resorting to the PCA technique. In particular, given a generic $n \times p$ data matrix $\boldsymbol{X}$ and let $\texttt{pca}(\boldsymbol{X})$ be the algorithm which returns the principal component coefficients[2],

[1] This can be done very easily since the c3d format provided by the mocap system sets a `NaN` for each coordinate not reconstructed.

[2] For example, in MATLAB© this algorithm corresponds to the function `princomp`.

then $\boldsymbol{R} = \texttt{pca}(\boldsymbol{X})$ is a $p \times p$ matrix each column containing coefficients for one principal component, ordered in descending order of variance [44]. Thus, the first column of $\boldsymbol{R}$ indicates the direction of maximum variance of the data and the last column indicates the direction of the minimum variance of the data.

With the aim of lowering the noise sensitivity of the estimated joint axis direction, a unique PCA is applied to all the markers moving when joint i is actuated. Before computing the PCA of the data, the paths have to be suitably translated so that, for a revolute joint, all the circumferences lay on a same plane passing through the origin, and, for a prismatic joint, all the straight lines pass through the origin. This is easily accomplished by subtracting to each column of the matrices $\widetilde{\boldsymbol{C}}_{ij}$ in (5.5) its mean, thus obtaining new matrices $\boldsymbol{C}_{ij}$.

For each robot joint, the PCA algorithm is then applied to the $(N \cdot M_i) \times 3$ matrix $\boldsymbol{C}_i$ constructed as

$$\boldsymbol{C}_i = \begin{pmatrix} \boldsymbol{C}_{i1} \\ \vdots \\ \boldsymbol{C}_{iM_i} \end{pmatrix}, \tag{5.6}$$

obtaining the rotation matrix[3]

$$\boldsymbol{R}_i = \texttt{pca}(\boldsymbol{C}_i), \tag{5.7}$$

which, for a revolute joint, expresses the orientation of a right-handed frame attached to the plane of the circular paths (translated as explained before and hereafter, called the "PCA plane"), with the z axis normal to the plane. Such frame will be hereafter called the "PCA frame".

Now, for a revolute joint, the joint axis is the rotation axis of the motions generating all the circular paths of the markers, hence the estimated direction of the joint i axis, i.e., according to the DH convention, the axis $\boldsymbol{z}_{i-1}$ of the frame fixed to link $i-1$, is the direction normal to the plane of the paths, i.e. the direction of minimum variance of the data that is the third column of the matrix $\boldsymbol{R}_i$. On the other hand, for a prismatic joint, the direction of the join axes is the direction of maximum variance of the data, i.e. the first column of $\boldsymbol{R}_i$.

In conclusion, the estimated joint i axis direction is

$$\boldsymbol{z}_{i-1} = \begin{cases} \boldsymbol{R}_i \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}} & \text{revolute joint} \\ \boldsymbol{R}_i \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathrm{T}} & \text{prismatic joint} \end{cases}, \quad i = 1, \ldots, n. \tag{5.8}$$

[3]The directions provided by the `pca` algorithm are assumed to constitute an orthonormal right-handed frame, hence $\boldsymbol{R}_i$ is a rotation matrix with $\det(\boldsymbol{R}_i) = 1$.

5.3.3 Identification of joint axes positions

In order to actually estimate the joint axes, besides their directions, a point belonging to each axis must be estimated. Of course, the position of these points with respect to the measurement frame depends on the robot configuration, therefore, it is hereafter assumed that the calibration motion of each joint always starts from the same robot configuration.

If joint i is prismatic, the actual location of the joint axis is not relevant, provided that the DH parameter a_i of link i is suitably redefined selecting the origin of the frame attached to link i on the axis of the first following revolute joint. In case the last joint is prismatic, then the joint axis will be selected passing through one of the markers attached to the last link, and the mean of its coordinates during the joint motion is considered as its estimated position.

For a revolute joint, the identification of the joint axis position can be accomplished by estimating the center of the circular paths in the PCA plane. This can be done by resorting to one of the numerous algorithms for circle fitting available in the literature, e.g. [24, 61, 72]. Given a generic $n \times 2$ data matrix $\boldsymbol{X}$ containing the 2D coordinates of n points laying on a plane to be fitted by a circumference, let $\texttt{circle\_fitting}(\boldsymbol{X})$ denote the algorithm which gives the coordinates of the center point of the best fitting circumference in a least square sense. Before applying the algorithm for each revolute joint i, the acquired marker positions in (5.5) have to be expressed in the PCA frame so that their z coordinate is constant (except the noise) and thus only the first two coordinates are used in the circle fitting procedure. The $N \times 3$ data matrices (one for each marker moving when joint i is actuated) containing the marker positions expressed in the PCA frame can be easily obtained as

$$^{\texttt{pca}}\widetilde{\boldsymbol{C}}_{ij} = \widetilde{\boldsymbol{C}}_{ij}\boldsymbol{R}_i, \quad j = 1, \ldots, M_i, \tag{5.9}$$

while the $N \times 2$ data matrices with the x and y coordinates only are

$$^{\texttt{xy}}\widetilde{\boldsymbol{C}}_{ij} = {}^{\texttt{pca}}\widetilde{\boldsymbol{C}}_{ij} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad j = 1, \ldots, M_i. \tag{5.10}$$

Then, for each marker j, the center of the best fitting circumference is obtained as

$$^{\texttt{xy}}\boldsymbol{c}_{ij} = \texttt{circle\_fitting}(^{\texttt{xy}}\widetilde{\boldsymbol{C}}_{ij}) \tag{5.11}$$

and its 3D coordinates in the measurement frame are obtained by using as z coordinate the mean $\bar{z}_{ij}$ of the third column of the matrix $^{\texttt{pca}}\widetilde{\boldsymbol{C}}_{ij}$, discarded

before in the circle fitting procedure, and by rotating the data back in the measurement frame through the rotation matrix $\boldsymbol{R}_i$, i.e.

$$\boldsymbol{c}_{ij} = \boldsymbol{R}_i \begin{pmatrix} {}^{\mathrm{xy}}\boldsymbol{c}_{ij} \\ \bar{z}_{ij} \end{pmatrix}. \tag{5.12}$$

Naturally, until now, for each revolute joint i, M_i positions of the joint axis have been estimated, one for each marker moving when the joint is actuated. As the actual location of the axis, the centre point estimated with the lower residual is selected, i.e.

$$\boldsymbol{c}_i = \boldsymbol{c}_{i\bar{\jmath}}, \text{ with } \quad \bar{\jmath} = \arg \min_{j=1,\ldots,M_i} MSE_j, \tag{5.13}$$

where MSE_j is the mean square error obtained with the `circle_fitting` algorithm used in (5.11). Alternative methods could be used which use all the markers at the same time, like the one proposed in [15] for determining in closed form the centre of rotation of a spherical joint, but suitably adapted for a rotational joint. However, such method is based on the computation of a QZ factorisation to solve a generalised eigenvalue problem and then on the selection of the smallest positive eigenvalue. In presence of noise, the smallest positive eigenvalue is not always the one associated to the best fit and thus the numerical implementation of the algorithm is problematic, since the choice of the right eigenvalue depends on the noise level.

5.3.4 Computation of the kinematic parameters

As said, the joint readings are assumed accurate enough since specific methods can be used to correct them, e.g. [17] and references therein. Therefore, only actual kinematic parameters will be computed based on their definitions given at the end of Section 5.1.

These definitions firstly require to identify the frames attached to each link. According to the DH kinematic model recalled in Section 5.1, the $\boldsymbol{z}_i$ axis is selected along the axis of joint $i+1$, the $\boldsymbol{x}_i$ axis is selected along the common normal to axes $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ computed by following the procedure outlined in 5.3.5 if the consecutive joint axes are parallel or not, and the procedure outlined in 5.3.6 if the axes are near parallel. The $\boldsymbol{y}_i$ axis is simply obtained to get a right-handed frame as $\boldsymbol{y}_i = \boldsymbol{x}_i \times \boldsymbol{z}_i$. The origin O_i of each frame is selected at the point of intersection of the common normal of axes $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ with the $\boldsymbol{z}_i$ axis, when the axes are non parallel. If the axes are parallel then the DH convention suggests to select the origin O_i so as to set $d_i = 0$ if joint i is revolute, otherwise O_i can be selected, e.g., at

the mechanical limit of the joint range. If they are near parallel the origin is selected according to (5.22) as explained in 5.3.6.

For each link i, connecting joints i and $i+1$, the computation of the parameter a_i can be done by computing the distance between the two lines with directions $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ obtained in Eq. (5.8) and passing through the points $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ obtained in (5.13), respectively. As discussed in the previous subsection, $\boldsymbol{c}_i$ is obtained as in Eq. (5.12) if joints i is revolute, or, if joint i is prismatic, as the location of the axis of the first following revolute joint. The distance between two lines can be obtained in several manners, here the algorithm reported in 5.3.5 is adopted. In particular, Eq. (5.18) is used for parallel axes, while Eq. (5.20) is used for non parallel axes. To handle the case of near parallel axes, the Hayati convention is used as explained in 5.3.6.

The parameter d_i is computed only for revolute joints and it set to zero both in the case the axes $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ are parallel or near parallel[4]. In case of non parallel axes, it is computed as the distance between the points of intersection of the common normals of two consecutive joints that can be easily found as specified in 5.3.5.

The parameter α_i, i.e. the angle about axis $\boldsymbol{x}_i$ between the joint axes $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$, can be computed by observing that from Eq. (5.2) the rotation matrix $\boldsymbol{R}_{i-1}^{i}$ expressing the orientation of frame $i-1$ with respect to frame i can be easily extracted as

$$\boldsymbol{R}_{i-1}^{i} = \begin{pmatrix} \boldsymbol{x}_{i-1}^{i} & \boldsymbol{y}_{i-1}^{i} & \boldsymbol{z}_{i-1}^{i} \end{pmatrix} = \begin{pmatrix} \cos\vartheta_i & \sin\vartheta_i & 0 \\ -\sin\vartheta_i\cos\alpha_i & \cos\vartheta_i\cos\alpha_i & \sin\alpha_i \\ \sin\vartheta_i\sin\alpha_i & -\cos\vartheta_i\sin\alpha_i & \cos\alpha_i \end{pmatrix}, \tag{5.14}$$

and thus, the angle α_i can be computed as[5]

$$\alpha_i = \mathrm{Atan2}(z_{i-1_y}, z_{i-1_z}), \tag{5.15}$$

where z_{i-1_y} and z_{i-1_z} are the second and third component of the vector $\boldsymbol{z}_{i-1}^{i}$ expressed in frame i. This vector can be obtained from the relationship

$$\boldsymbol{z}_{i-1}^{i} = \boldsymbol{R}_i^{\mathrm{T}}\boldsymbol{z}_{i-1} = \begin{pmatrix} \boldsymbol{x}_i & \boldsymbol{y}_i & \boldsymbol{z}_i \end{pmatrix}^{\mathrm{T}} \boldsymbol{z}_{i-1}$$

where the axis $\boldsymbol{z}_{i-1}$ is estimated in (5.8) and the rotation matrix $\boldsymbol{R}_i$ expressing the orientation of frame i with respect to the base frame is known once the axes of the frame attached to link i have been estimated as explained at the beginning of this section.

[4]This is established as explained in 5.3.5.

[5]The function $\mathrm{Atan2}(b, a)$ is defined as the phase of the complex number $a + ib$.

The parameter ϑ_i, computed only for prismatic joints as the angle between the axes $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_i$ about axis $\boldsymbol{z}_{i-1}$, is determined in a similar way. By inspecting the rotation matrix in (5.2), the angle ϑ_i can be computed as

$$\vartheta_i = \text{Atan2}(x_{i_y}, x_{i_x}), \tag{5.16}$$

where x_{i_y} and x_{i_x} are the second and first component of the vector $\boldsymbol{x}_i^{i-1}$ expressed in frame $i-1$. This vector can be obtained from the relationship

$$\boldsymbol{x}_i^{i-1} = \boldsymbol{R}_{i-1}^{\mathrm{T}} \boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{x}_{i-1} & \boldsymbol{y}_{i-1} & \boldsymbol{z}_{i-1} \end{pmatrix}^{\mathrm{T}} \boldsymbol{x}_i,$$

where all the axes are known once the axes of the frames attached to links have been estimated as explained at the beginning of this section. Actually, the determination of the angle ϑ_i can be carried out also for revolute joints and this allows to calibrate also the offsets of the joint angular sensors. It is sufficient to place the robot in a given configuration and move one joint at a time of a given angular displacement, the difference between the computed ϑ and the commanded displacement is the sought offset.

The additional parameter β_i, foreseen by the Hayati convention [40], can be computed, for a revolute joint, by taking the transpose of the rotation matrix in (5.3) and by inspection it is easy to derive that

$$\beta_i = \text{Atan2}(-z_{i_x}, z_{i_z}), \tag{5.17}$$

where z_{i_x} and z_{i_y} are the second and first component of the vector $\boldsymbol{z}_i^i$ expressed in frame i. This vector can be obtained from the relationship

$$\boldsymbol{z}_i^i = \boldsymbol{R}_i^{\mathrm{T}} \boldsymbol{z}_i = \begin{pmatrix} \boldsymbol{x}_i & \boldsymbol{y}_i & \boldsymbol{z}_i \end{pmatrix}^{\mathrm{T}} \boldsymbol{z}_i,$$

where all the axes are known once the axes of the frames attached to links have been estimated as explained at the beginning of this section. The same procedure can be applied for a prismatic joint taking into account the transpose of the rotation matrix in (5.4).

5.3.5 Common normal and distance between two lines

The following algorithm for the computation of the distance between two lines is used to compute the a_i DH parameters. Identifying a line by the couples of the unit vector $\boldsymbol{v}$ of its direction and a point with position vector $\boldsymbol{p}$, the distance of two lines $\{\boldsymbol{v}_1, \boldsymbol{p}_1\}$ and $\{\boldsymbol{v}_2, \boldsymbol{p}_2\}$ is computed in different ways if the lines are parallel or not. To decide if they are parallel or not, the following measure is computed

$$a = ||\boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_2| - 1|$$

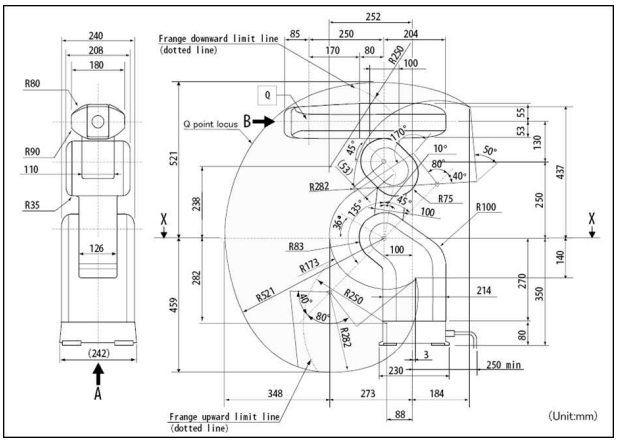


Figure 5.3: CAD drawing of the Mitsubishi Melfa RV-2A industrial arm used in both simulation and experiments.

and compared to the two thresholds $a_1 = 10^{-6}$ and $a_2 = 10^{-3}$ selected by carrying out a number of numerical simulations.

If $a < a_1$ then the lines are parallel and the distance is computed as

$$d = \|\boldsymbol{p}_2 - \boldsymbol{p}_1 + (\boldsymbol{p}_1 - \boldsymbol{p}_2)^{\mathrm{T}}\boldsymbol{v}_1\boldsymbol{v}_1\|. \tag{5.18}$$

In this case, the direction of the common normal is uniquely defined (while its location is not) as

$$\boldsymbol{n} = (\boldsymbol{p}_2 - \boldsymbol{p}_1 + (\boldsymbol{p}_1 - \boldsymbol{p}_2)^{\mathrm{T}}\boldsymbol{v}_1\boldsymbol{v}_1)/d. \tag{5.19}$$

If $a > a_2$ then the lines are not parallel and the distance is computed as

$$d = \|\boldsymbol{p}_1 + t_1\boldsymbol{v}_1 - \boldsymbol{p}_2 - t_2\boldsymbol{v}_2\| \tag{5.20}$$

where the vector $\boldsymbol{t} = (t_1 \quad t_2)^{\mathrm{T}}$ is obtained as

$$\boldsymbol{t} = \begin{pmatrix} -1 & \boldsymbol{v}_1^{\mathrm{T}}\boldsymbol{v}_2 \\ -\boldsymbol{v}_1^{\mathrm{T}}\boldsymbol{v}_2 & 1 \end{pmatrix}^{-1} \begin{pmatrix} (\boldsymbol{p}_1 - \boldsymbol{p}_2)^{\mathrm{T}}\boldsymbol{v}_1 \\ (\boldsymbol{p}_1 - \boldsymbol{p}_2)^{\mathrm{T}}\boldsymbol{v}_2 \end{pmatrix}.$$

Note that the points with positions $p_1 + t_1\boldsymbol{v}_1$ and $p_2 + t_2\boldsymbol{v}_2$ are the intersections of the lines with the common normal of the two lines. This allows to obtain also the direction of the common normal to two lines, as the difference of this two points, namely

$$\boldsymbol{n} = (\boldsymbol{p}_1 + t_1\boldsymbol{v}_1 - \boldsymbol{p}_2 - t_2\boldsymbol{v}_2)/d. \tag{5.21}$$

Figure 5.4: Picture of the Mitsubishi Melfa RV-2A industrial arm used in both simulation and experiments (three cameras and reflective markers attached to the robot links are also visible).

If $a_1 \leq a \leq a_2$ then the lines are considered near parallel and the DH parameter a_i is not computed as the distance between the two lines but according to the Hayati convention.

5.3.6 Hayati convention

If joint axes $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ are near parallel, the direction of axis $\boldsymbol{x}_i$ of the frame fixed to link i is selected as the line passing through the origin O_{i-1} of frame $i-1$ and the point of intersection O_i of axis $\boldsymbol{z}_i$ and the plane orthogonal to axis $\boldsymbol{z}_{i-1}$ passing through the origin O_{i-1} of frame $i-1$ (see Fig. 5.5). To compute the position of this point of intersection, the following algorithm can be used.

Using the same notation of 5.3.5, given two lines $\{\boldsymbol{v}_1, \boldsymbol{p}_1\}$ and $\{\boldsymbol{v}_2, \boldsymbol{p}_2\}$, a point with position $\boldsymbol{p}$ on the plane passing through the point with position $\boldsymbol{p}_1$ and orthogonal to the first line is such that

$$(\boldsymbol{p} - \boldsymbol{p}_1)^{\mathrm{T}} \boldsymbol{v}_1 = 0.$$

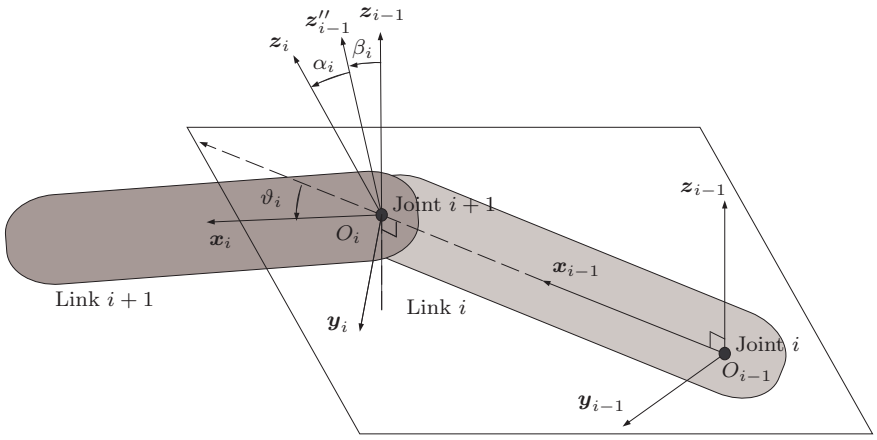


Figure 5.5: Hayati convention for near parallel revolute joints.

The point with position $\boldsymbol{p}$ belongs also to the second line if

$$(\boldsymbol{p}_2 + t\boldsymbol{v}_2)^{\mathrm{T}}\boldsymbol{v}_1 = 0 \;\Rightarrow\; t = \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)^{\mathrm{T}}\boldsymbol{v}_1}{\boldsymbol{v}_2^{\mathrm{T}}\boldsymbol{v}_1},$$

where t is the parameter of the parametric equation of the second line $\boldsymbol{p} = \boldsymbol{p}_2 + t\boldsymbol{v}_2$. Therefore, the sought point of intersection has position

$$\boldsymbol{p}_2 + \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)^{\mathrm{T}}\boldsymbol{v}_1}{\boldsymbol{v}_2^{\mathrm{T}}\boldsymbol{v}_1}\boldsymbol{v}_2 \tag{5.22}$$

Of course, the expression above is meaningful only if the two lines are not orthogonal, and this is certainly true in case the Hayati convention has to be used, i.e. the two lines are near parallel. In such a case, the $\boldsymbol{x}_i$ axis is selected as

$$\boldsymbol{x}_i = \frac{\boldsymbol{o}_i - \boldsymbol{o}_{i-1}}{\|\boldsymbol{o}_i - \boldsymbol{o}_{i-1}\|},$$

where $\boldsymbol{o}_i$, computed using Eq. (5.22), and $\boldsymbol{o}_{i-1}$ are the position vectors of the points O_i and O_{i-1} defined above, respectively.

5.4 Simulations

In order to verify that the algorithm is actually able to correctly estimate the kinematic parameters of an industrial manipulator, a simulation study has been performed by means of the MATLAB© Robotics Toolbox [22]. The kinematic model of a Mitsubishi Melfa RV-2A industrial arm has been

defined. Fig. 5.3 reports the CAD drawing of the arm, which is the same that will be experimentally calibrated as described in Section 5.5 and depicted in Fig. 5.4, where also three of the four cameras used for the calibration are visible together with some of the reflective markers attached to the robot links.

The purpose of the first simulation case study is to show that the identification procedure is *correct*, in the sense that it is able to produce the true kinematic parameters if the measurements are not affected by noise. Of course, in simulation the true kinematic parameters to be correctly identified are those used in the model exploited to fictitiously generate the measurements, namely the nominal one reported in Tab. 5.2.

Link	a_i [m]	α_i [deg]	d_i [m]	ϑ_i [deg]
1	0.100	90.0	0.000	q_1
2	0.250	0.000	0.000	q_2
3	0.130	90.0	0.000	q_3
4	0.000	−90.0	0.250	q_4
5	0.000	90.0	0.000	q_5
6	0.000	0.000	0.085	q_6

Table 5.2: Nominal kinematic parameters of the Melfa RV-2A arm.

Once the correctness will be proved, a second simulation case study will be carried out to quantify the effects of measurement noise on the proposed calibration algorithm.

The marker trajectories (see Fig. 5.6 for an example) have been generated by assuming one marker attached to each link and by placing the base frame of the robot in a location different from the location of the frame 0 so as to simulate that the 3D marker positions are acquired with respect to a measurement frame other than the frame 0 of the robot. Moreover, the initial configuration of the robot has been selected equal to the one that will be used in the experiments and chosen so as to maximise the range of motions of the robot joints seen by the cameras, i.e.

$$\boldsymbol{q}_i = [-41.0 \quad 64.0 \quad 74.0 \quad 14.0 \quad -6.0 \quad -3.0]^{\mathrm{T}} \,\mathrm{deg}.$$

By following the four steps of the procedure described in Section 5.3, the kinematic parameters have been correctly identified with an error of the order of the floating point double precision. In particular, the data acquisition performed through the motion capture system has been substituted by the synthetic generation of the 3D marker positions. The results of the third

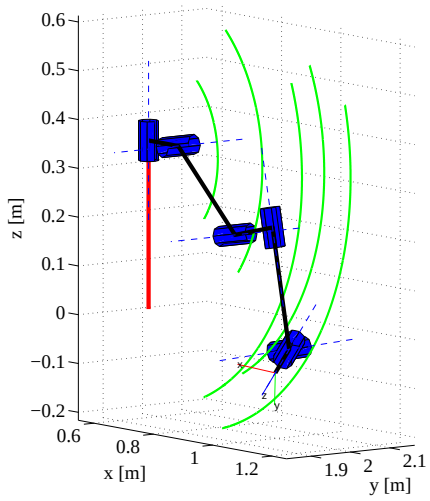


Figure 5.6: Simulated marker trajectories without measurement noise (joint 2 moving).

intermediate step involving the circle fitting of the marker trajectories projected on the PCA xy plane, identified in the second step, are reported in Fig. 5.7. The mean square errors MSE_j in 5.13 are all of the order 10^{-27}, the floating point double precision. Note that for each joint only the estimated circumference corresponding to the minimum square error has been displayed.

In the second case study, a zero mean Gaussian noise with a standard deviation of 10^{-3} m, much larger than the typical standard deviation of the noise affecting optical motion capture system, has been added to the marker trajectories in all directions. The resulting errors on the identified parameters are reported in Tab. 5.3. It is evident how the algorithm is able to effectively counteract the presence of noise since the obtained errors are significantly lower than the standard deviation of the measurement noise. This can be ascribed to the noise reducing effect of both the PCA algorithm and the least square circle fitting algorithm.

Notably, the angle between joint axes 2 and 3 has been estimated close enough to zero such that the axes have been considered parallel and thus the Hayati convention has not been used. Therefore, to test also this part of the algorithm an additional case study has been carried out by artificially modifying the angle α_2 setting it to the value 0.57 deg with a null β_2 angle. In this case, the algorithm estimated the angles $\alpha_2 = 0.58$ deg and $\beta_2 = 0$ deg, again with a small error.

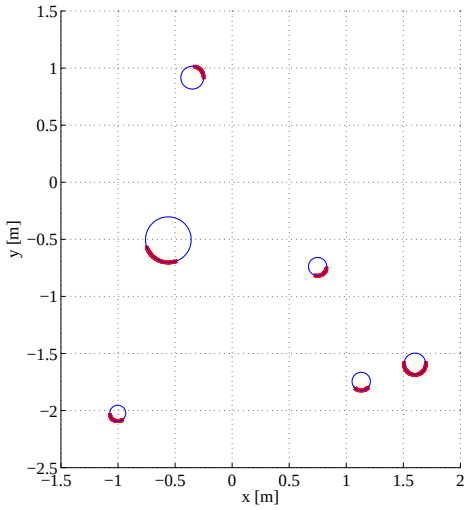


Figure 5.7: Estimated circumferences on the PCA plane in the simulated calibration.

Link	Δa_i [mm]	$\Delta\alpha_i$ [deg]	Δd_i [mm]
1	0.2	0.0000	0.0
2	0.2	0.0057	0.0
3	0.1	0.0172	0.0
4	0.1	0.0115	0.2
5	0.1	0.0115	0.1
6	0.0	0.0000	0.0

Table 5.3: Identification errors obtained in the second simulation case study (measurement noise with 10^{-3} m standard deviation).

5.5 Experimental results

For the experimental kinematic calibration, the initial configuration of the robot has been selected equal to the one used in the simulation, the range of motion of each joint is reported in Tab. 5.4. The robot has been programmed to move one joint at a time with a 4% of speed override, which led to capturing 6000 samples, and only about 3000 of which have been used after removing all the samples where not all markers are visible.

The markers have been attached to the robot links as shown in Fig. 5.4 and a total of 11 markers have been used, two for each link except for the last link where only one marker has been attached. Once the data (3D marker positions) have been acquired, the second and third steps of the pro-

Joint	$q_{\min}$ [deg]	$q_{\max}$ [deg]
1	−43.0	125.0
2	2.0	92.0
3	52.0	150.0
4	−56.0	160.0
5	−40.0	80.0
6	−54.0	155.0

Table 5.4: Range of motion of the robot joints for the experimental calibration.

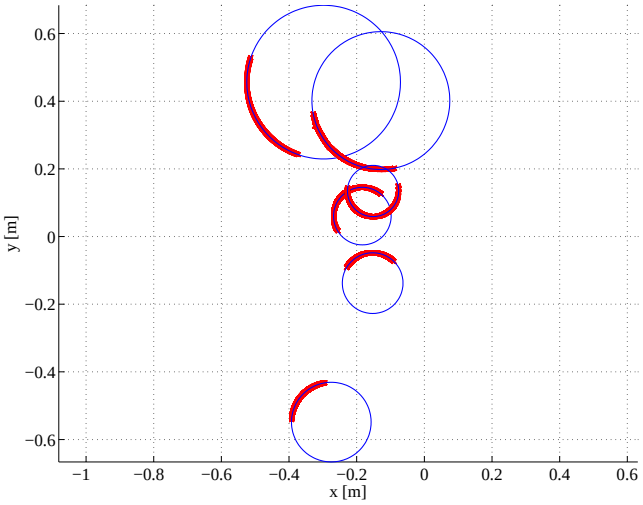


Figure 5.8: Estimated circumferences on the PCA plane in the experimental calibration.

cedure have been applied to estimate the joint axes directions and locations. Figure 5.8 shows the estimated circumferences best fitting the marker trajectories projected on the PCA plane. As done for the simulation results, only one circumference for each joint axis has been plotted and the obtained mean square errors are reported in Tab. 5.5. The last step of the procedure led to the estimated kinematic parameters reported in Tab. 5.6. Of course, in the experimental phase it is not possible to explicitly compute the estimation errors of the parameters, therefore an additional experiment has been carried out to show how the use of the estimated kinematic parameters in lieu of the nominal ones can significantly improve the positioning accuracy of the robot. To this aim, the position of the robot end-effector has been acquired still using the motion capture system in 20 different locations and

Joint	MSE [m]
1	$3.0 \cdot 10^{-9}$
2	$3.4 \cdot 10^{-9}$
3	$3.7 \cdot 10^{-7}$
4	$2.4 \cdot 10^{-8}$
5	$1.7 \cdot 10^{-8}$
6	$3.5 \cdot 10^{-8}$

Table 5.5: Mean square errors of the circle fitting algorithm for the experimental calibration.

it has been compared to the position computed using both the nominal and the estimated kinematic parameters. The obtained errors are reported in Fig. 5.9, where an average improvement of about 50% of the accuracy has been obtained with the estimated kinematic parameters.

Link	a_i [m]	α_i [deg]	d_i [m]
1	0.0995	89.957	0.0000
2	0.2545	0.1590	0.0000
3	0.1259	89.887	0.0013
4	0.0007	−89.724	0.2492
5	0.0010	89.799	0.0004
6	0.0000	0.0000	0.0983

Table 5.6: Calibrated kinematic parameters.

5.6 Conclusions

The proposed calibration procedure for estimating the kinematic parameters of an serial manipulator adopts a measurement system allowing a 3D position measurement distributed along the entire kinematic chain not only of its end effector. This novel measurement process enables the estimation of the kinematic parameters in a closed form without the need to resort to any linearisation of the error function. Another characteristic of the proposed algorithm is its robustness against measurement noise, which is guaranteed by the adoption of the PCA method to estimate the directions of the robot joint axes. The kinematic calibration procedure, easy to set up and fast to carry out, appears suitable for robot manufactures, who can use the estimated kinematic parameters in the control units of the robot for the computation of both direct and inverse kinematics to improve the positioning accuracy of

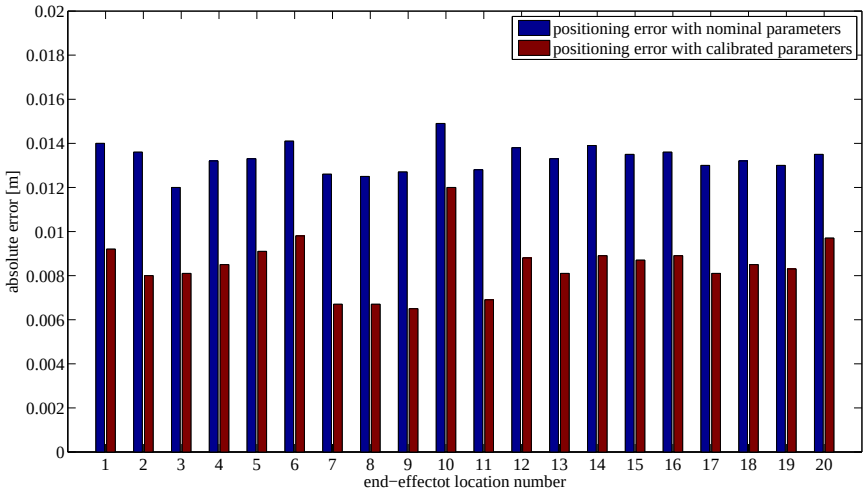


Figure 5.9: Absolute accuracy improvement after the experimental calibration.

the end effector. Both simulation and experimental results obtained in the calibration of an industrial 6DOF robot demonstrated the effectiveness of the proposed approach in improving the positioning accuracy of the manipulator.

Chapter 6

Conclusions and future work

In this work, a complex hardware/software architecture has been proposed, that has the very ambitious objective to solve the problem of the observation of human manipulation. With the diffusion of anthropomorphic robotic systems provided with human-like hands, this problem has become very relevant and the request of accurate observation systems at finger level has been increasing more and more. On the other hand, methods in literature and commercial system do not offer effective solutions for this issue. The methods proposed and developed in this work can be adopted by the robotics and virtual reality experts to perform an accurate and reliable hand observation, both for research applications and for developing a new generation of commercial systems. Even if the experimental result are really encouraging, some open problems still remain and possible improvements can be investigated. At architectural level, an interesting theoretical problem is the quantitative evaluation of the measurement error made by the system in the estimation of hand joint angles, since a quantitative ground truth is not available. A mathematical study could be carried out to obtain a theoretical estimation of the error probability density functions after the refinement applied by the high-level sensor fusion module.

Brief considerations about conclusions and possible improvements at single component level are discussed in Sections 6.1, 6.2, 6.3.

6.1 Sensing system

A low-cost data glove prototype has been presented, whose performances are comparable with the more complex and expensive commercial models. The data glove is particularly suitable in measurement systems for observing the human manipulation based on sensor fusion. Future work will consist in im-

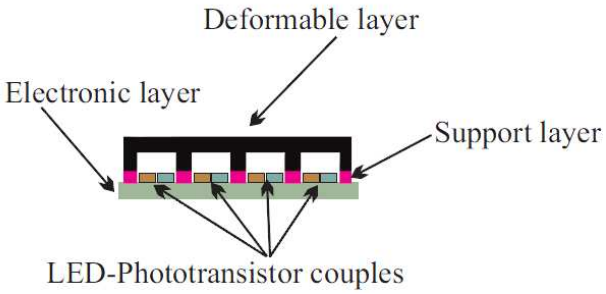


Figure 6.1: Tactile sensor concept based on LED-phototransistor couples

proving the installation procedure of the sensing elements and of introducing additional conditioning electronics, in order to obtain a calibration curves less dependent from the performer and to evaluate the data glove performance in a more rigorous mode, for example as proposed in [32] and in [78].

Another absolutely significative advancement would consist in providing the data-glove with novel tactile sensors developed at Automatic Control Laboratory of Second University of Naples, instead of the commercial ones used in this work. The objective of such sensors is to provide information about the contact point/area between the fingertips and the manipulated object, together with an estimation of both the normal and tangential components of the contact force. The direction of the contact forces is often neglected , but it can bring several advantages in the accurate observation of human manipulation as well as in control of anthropomorphic robotic hands. The novel tactile sensors are based on the use of LED-phototransistor couples and a deformable elastic layer positioned above the optoelectronics devices (see Fig. 6.1). For more details about the novel developed tactile sensors see [26].

In order to have cheaper but relatively accurate observation systems, lower-cost hardware able to measure the positions of markers in space might be considered. An interesting issue would be, for example, to investigate the possibility of adopting the new commercial device Kinect [3] as an input of the low-level sensor fusion module instead of the motion capture system or the possibility to include it within the proposed architecture as a third kinematic data source.

6.2 Low-level module

In order to obtain a more accurate estimation of joint angles, more advanced Bayesian sensor fusion methods could be evaluated, e.g. Unscented Kalman Filter (UKF) and Particle Filters (PF). The UKF presents the same assumptions as the EKF but seems to have better performances in terms of accuracy and execution time. PF, on the other hand, offers the interesting possibility to consider non-Gaussian noise both in the state-update function and in the measurement-state relation.

6.3 High-level module

A method to improve the human motion observation (i.e. estimation of the hand pose in the space and of joint angles), exploiting the fingertip contact forces measurements, has been presented. Since a Jacobian-based IK method has been adopted, it is possible to apply the correction on-line. The measurements from the sensing system are aimed at giving the correction algorithm a good starting point. Then, the better the sensing system, the faster the correction and the more "human like" the corrected posture are expected to be. The future work will be aimed at two objectives. The first objective will be to add other features, keeping the possibility of an on-line execution. For example, an interesting challenges to afford would be the definition of methods more robust to noise in the force measurements to decide if a finger is in contact with the object (e.g. based on probabilistic theory or fuzzy logic). The second objective will consist in exploiting the novel tactile sensors described in Section 6.1. The knowledge of contact points and the components of the forces would allow, if used in a smart way, a more precise correction of the hand posture and an extended definition of kinetostatic consistency.

Bibliography

[1] http://www.c3d.org.

[2] Dexmart project. http://www.dexmart.eu.

[3] Kinect device. http://www.xbox.com/kinect.

[4] Pressure Profile: FingerTPS.

[5] PCT WO 2004/059249 A1. Goniometric sensor. *International Patent Application*, 15 July 2004.

[6] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. 25(5):985–994, 2009.

[7] G. Antonelli, F. Arrichiello, and . Chiaverini. The nsb control: a behavior-based approach for multi-robot systems. *PALADYN Journal of Behavioral Robotics*, 1(1):48–56, 2010.

[8] P. Azad, T. Gockel, and R. Dillmann. *Computer Vision – das Praxisbuch*. Elektor-Verlag, 2007.

[9] L.K. Barker. Vector-algebra approach to extract denavit-hartenberg parameters of assembled robot arms. *NASA Technology Paper*, 2191, August 1983.

[10] B. Benhabib, A.A. Goldenberg, and R.G. Fenton. A solution to the inverse kinematics of redundant manipulators. In *Proc. 1985 American Control Conference*, Boston, MA, 1985.

[11] A. Cavallo, G. De Maria, C. Natale, and S. Pirozzi. Optoelectronic joint angular sensor for robotic fingers. *Sensors and Actuators A: Physical*, 152(2):203–210, 2009.

[12] P. Cerveri, N. Lopomo, A. Pedotti, and G.Ferrigno. Derivation of centers and axes of rotation for wrist and finger in a hand kinematic model: Methods and reliability results. *Annals of Biomechanics Enginnering*, 33(3):402–412, 2005.

[13] P. Cerveri, A. Pedotti, and G. Ferrigno. Robust recovery of human motion from video using kalman filters and virtual humans. *Human Movement Science*, (22):377–404, 2003.

[14] H. Chanala, E. Duca, P.Raya, and J.Y. Hascoe. A new approach for the geometrical calibration of parallel kinematics machines tools based on the machining of a dedicated part. *International Journal of Machine Tools & Manufacture*, 47:1151–1163, 2007.

[15] L.Y. Chang and N.S. Pollard. Constrained least-squares optimization for robust estimation of center of rotation. *Journal of Biomechanics*, 40:1392–1400, 2007.

[16] L.Y. Chang and N.S. Pollard. Method for determining kinematic parameters of the in vivo thumb carpometacarpal joint. *IEEE Transactions on Biomedical Engineering*, 55(7):1897 – 1906, 2008.

[17] Heping Chen, Thomas Fuhlbrigge, Sang Choi, Jianjun Wang, and Xiongzi Li. Practical industrial robot zero offset calibration. In *4th IEEE Conference on Automation Science and Engineering*, pages 516–521, Washington DC, USA, August 2008.

[18] I-M. Chen, G. Yang, C.T. Tan, and S.H. Yeo. Local POE model for robot kinematic calibration. *Mechanism and Machine Theory*, 36:1215–1239, 2001.

[19] S Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. 13(3):398–410, 1997.

[20] Dexmart Consortium. Sensor fusion of kinetostatic data. *Internal Report I1.3*, 2010.

[21] F. Corato, P. Falco, M. Lösch, E. Maggio, R. Jäkel, and L. Villani. Original approaches to interpretation, learning and modeling, from the observation of human manipulation. In *Conference on Robotics Science and Systems (RSS Conference)*, Seattle, June 2009.

[22] P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996.

[23] C. Corral and C. Lindquist. On implementing kasas circle fit procedure. *IEEE Transactions on Instrumentation and Measurement*, 47(3):789–795, 1998.

[24] C.A. Corral and C.S. Lindquist. On implementing Kasa's circle fit procedure. *IEEE Transactions on Instrumentation and Measurement*, 47(3):789–795, 1998.

[25] A. Steinberg D. Hall. Dirty secrets in multisensor data fusion. In *National Symposium on Sensor Data Fusion*, San Antonio, 2000.

[26] A. D'Amore, G. De Maria, L. Grassia, C. NATALE, and S. Pirozzi. Silicone rubber based tactile sensor for measurement of normal and tangential components of the contact force. *Journal of Applied Polymer Science, Wiley Periodicals, Inc.,*, 122:3758–3770, 2011.

[27] H. Das, J-J.E. Slotine, and T.B. Sheridan. Inverse kinematic algorithms for redundant systems. In *Proc. IEEE International Conference on Robotics and Automations, (ICRA'88)*, pages 43–48, San Francisco, CA, 1988.

[28] G. De Maria and R. Marino. A discrete algorithm for solving the inverse kinematic problem for robotic manipulators. In *Proc. 2nd International Conference on Advanced Robotics*, pages 275–282, Tokyo, Japan, 1985.

[29] J. Denavit and R.S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77:215–221, 1955.

[30] J. Denavit and R.S. Hartneberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, pages 215–221, 1955.

[31] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, August 2010.

[32] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario. Evaluation of an instrumented glove for hand-movement acquisition. *Journal of Rehabilitation Research and Development*, 40(2):179–190, 2003.

[33] K. Dorfmuller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. In *IEEE/AC International Symposium on Augmented Reality*, 2001.

[34] Staffan Ekvall and Danica Kragic. Interactive grasp learning based on human demonstration. In *ICRA*, pages 3519–3524. IEEE, 2004.

[35] P. Falco and C. Natale. A sensor fusion approach to observation of human hand motion. In *1st International Conference on Applied Bionics and Biomechanics ICABB-2010*, Venezia, Italy, October 2010.

[36] P. Falco and C. Natale. On the stability of closed-loop inverse kinematics algorithms for redundant robots. *IEEE Transactions on Robotics*, 27(4), 2011.

[37] Michele Folgheraiter, Ilario Baragiola, and Giuseppina C. Gini. Teaching grasping to a humanoid hand as a generalization of human grasping data. In Jesús A. López, Emilio Benfenati, and Werner Dubitzky, editors, *KELSI*, volume 3303 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2004.

[38] S.S. H.U. Gamage and J. Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of Biomechanics*, 35:87–93, 2002.

[39] A.A. Goldenberg, B. Benhabib, and R.G. Fenton. A complete generalized solution to the inverse kinematics of robots. RA-I(1):14–20, 1985.

[40] S. Hayati and M. Mirmirani. Improving the absolute positioning accuracy of robot manipulators. *Journal of Robotic System*, 2(4):397–413, 1985.

[41] Ruibo He, Yingjun Zhao, Shunian Yang, and Shuzi Yang. Kinematic-parameter identification for serial-robot calibration based on POE formula. *IEEE Transactions on Robotics*, 26(3):411–423, 2010.

[42] J.K. Hollerbach and C.W. Wampler. The calibration index and the role of input noise in robot calibration. In G. Giralt and G Hirzinger, editors, *The Seventh International Symposium on Robotics Research*, pages 558–568. Springer-Verlag, London, 1996.

[43] J.M. Hollerbach and C.W. Wampler. The calibration index and a taxonomy for robot kinematic calibration methods. *Int. J. of Robotics Research*, 15(6):573–591, 1996.

[44] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, 2nd edition, 2002.

[45] S. Jung and K. Wohn. Tracking and motion estimation of the articulated object: a hierarchical kalman filter approach. *Real-Time Imaging 3*, (3):415–432, 1997.

[46] S. O. Kasap. *Optoelectronics and Photonics: Principles and Practices*. Prentice Hall, Englewood Cliffs, NJ, 2001.

[47] Alexander Kasper. KIT ObjectModels Web Database.

[48] V. Lakshmikantham and D. Trigiante. *Theory of Difference Equations*. Marcel Dekker, New York, NY, 2002.

[49] S. Lang. *Calculus of Several Variables*. Springer, New York, NY, 3rd edition, 2006.

[50] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes,. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3719–3726, 2000.

[51] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. 12:868–871, 1977.

[52] D.L.Hall ; S.A.H. McMullen. *Mathematical Techniques in Multisensor Data Fusion*. Artech House Publisher, Boston-London, second edition edition, 2004.

[53] N. Miyata, M. Kouhci, T. Kurihara, and M. Mochimaru. Modeling of human hand link structure from optical motion capture data. In *IEEE/RSJ International Conference on Intelligence Robots and Systems*, Sendai, Japan, September 28-October 2 2004.

[54] P.J. Nahin and J.L. Pokoski. Nctr plus sensor fusion equals iffn. In *IEEE Trans. Aerospace Electronic Systems*, San Antonio, 1980.

[55] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *Int. J. Robot. Res*, 6(2):3–15, 1987.

[56] C. Natale. Kinematic control of robots with noisy guidance systems. In *The 18th IFAC World Congress*, Milan, 2011.

[57] Erhan Oztop, Li-Heng Lin, Mitsuo Kawato, and Gordon Cheng. Extensive human training for robot skill synthesis: Validation on a robotic hand. In *ICRA*, pages 1788–1793. IEEE, 2007.

[58] G. Palli and S. Pirozzi. Force sensor based on discrete optoelectronic components and compliant frames. *Sensors and Actuators A: Physical*, 165:239–249, 2011.

[59] G. Palli and S. Pirozzi. Miniaturized optical-based force sensors for tendon-driven robots. In *Proc. Int. Conf. on Robotics and Automation*, pages 5344–5349, Shanghai, 2011.

[60] E.P. Pitarch, J. Yang, and K. Abdel-Malek. Santos hand: A 25 degree-of-freedom model. In *Digital Human Modeling for Design and Engineering Symposium*, Iowa City, June 2005.

[61] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21:1115–1138, 1987.

[62] Xiao-Dong Ren, Zu-Ren Feng, and Cheng-Ping Su. A new calibration method for parallel kinematics machine tools using orientation constraint. *International Journal of Machine Tools & Manufacture*, 49:708–721, 2009.

[63] R. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter*. Artech House Publishers, Boston - London, 2004.

[64] D. Rotondo. A fuzzy system approach for estimation of joint angles of robotic and human fingers. Degree thesis, 2008.

[65] L. Sciavicco and B. Siciliano. Coordinate transformation: A solution algorithm for one class of robots. 16:550–559, 1986.

[66] B. Siciliano. A closed-loop inverse kinematic scheme for on-line joint-based robot control. *Robotica*, 8:231–243, 1990.

[67] B. Siciliano, L. Villani, G. Oriolo, and L. Sciavicco. *Robotics*. Springer, New York, NY, 2008.

[68] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics*. Springer Series in Advanced Textbooks in Control and Signal Processing. Springer, London, 1st edition, 2009.

[69] M.E. Sklar. Metrology and calibration techniques for the performance enhancement of industrial robots. In *Second Conference on Recent Advances in Robotics*, pages 178–202, Florida Atlantic University, FL, May 1988.

[70] H.W. Stone and A.C. Sanderson. A prototype arm signature identification system. In *1987 IEEE International Conference on Robotics and Automation*, pages 175–182, Raleigh, NC, May 1987.

[71] M. Tarokh and M. Kim. Inverse kinematics of 7-dof robots and limbs by decomposition and approximation. 23(3):595–600, 2007.

[72] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1115–1138, 1991.

[73] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[74] Jochen Triesch, Jan Wieghardt, Eric Maël, and Christoph von der Malsburg. Towards imitation learning of grasping movements by an autonomous robot. In *Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction*, GW '99, pages 73–84, London, UK, 1999. Springer-Verlag.

[75] M. Veber, T. Bajd, and M.Munih. Assessing joint angles in human hand via optical tracking device and calibrating instrumented glove. *Meccanica*, (42):451–463, 2007.

[76] K. Waldron and J. Schmiedeler. Kinematics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 1, pages 9–33. Springer, Heidelberg, 2008.

[77] C.W. Wampler, J.M. Hollerbach, and T. Arai. An implicit loop method for kinematic calibration and its application to closed-chain mechanisms. *IEEE Trans. Robotics and Automation*, 11(5):710–724, 1995.

[78] S Wise, W Gardner, E Sabelman, E Valainis, Y Wong, K Glass, J Drace, and J Rosen. Evaluation of a fiber optic glove for semi-automated goniometric measurements. *J Rehabil Res Dev*, pages 411–424, 1990.

[79] X. Zhang, S.W. Lee, and P. Braido. Determining finger segmental centers of rotation in flexion-extension based on surface marker measurement. *Journal of Biomechanics*, 36:1097–1102, 2003.

[80] J. Zhao and N. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *Trans. Comput. Graph.*, 13(4):313–336, 1994.

[81] H. Zhuang, K. Wang, and Z.S. Roth. Optimal selection of measurement configurations for robot calibration using simulated annealing. In *1994 IEEE Int Conf Robotics Automat*, pages 393–398, San Diego, CA, May 1994.

[82] R. Zöllner, O. Rogalla, and R. Dillmann. Integration of tactile sensors in a programming by demonstration system. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2578–2583, 2005.