POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Automation and Control Engineering

POLITECNICO
MILANO 1863

Human pose estimation in case of occlusions for safe and productive Human-Robot Interaction

Supervisor: Prof. Paolo Rocco

Co-supervisors: Andrea Casalino

Andrea Maria Zanchettin

Master Thesis dissertation of:

Sebastian Guzman Id. 859004

Academic year 2016-2017

Alla mia famiglia

Ringraziamenti

Un doveroso grazie è rivolto alla mia famiglia per avermi sempre sostenuto nel corso dei miei studi.
Ringrazio poi i miei compagni Fabio Allevi, Davide Bazzi, Stefano Dattilo, Giulio Dell'Oro, Luca Pozzessere e Dario Savaresi per questi anni passati insieme.
Infine ringrazio il Prof. Paolo Rocco, il Prof. Andrea Maria Zanchettin e Andrea Casalino per avermi dato l'opportunità di lavorare nell'ambito della robotica ed avermi aiutato nel corso di questi mesi.

Contents

List of Figures

List of Tables

Abstract

Collaborative robotics over the last few years has gained increasing interest in the industrial scenario. Collaborative robots, also called cobots, are equipped with sensors that allow them to perceive the surrounding environment. In particular, with regard to the synchronization of operations carried out by human and robot, a key role is taken by vision sensors. These sensors allow the robot to figure out where the human is at a certain time, to understand what task he is doing and even to predict his future actions.

At the basis of such processes, there is the human pose estimation algorithm. Estimating pose by an image is a difficult task for a number of reasons: variability of human visual appearance in images, variability in lighting conditions, variability in human aspect, partial occlusions, complexity of human skeletal structure and the loss of 3D information that results from observing the pose from 2D planar image projections, etc.. This problem has been approached using several techniques, but the one most used to estimate the human pose is for sure the Kalman filter. However, this technique presents some weaknesses, in particular, it is complex to take into account for occlusions in the estimation process.

This work aims to provide a solution to the problem of pose estimation of the operator in the situations of partial occlusion. The estimation algorithm proposed here is based on the particle filter technique which, as we will see, allows to introduce constraints in the estimation process that take into account possible occlusions. The algorithm has been implemented to use the depth data

provided by the Kinect V2. Finally, the results obtained have been compared to those obtained by applying the Kalman filter.

Sommario

La robotica collaborativa nel corso degli ultimi anni ha acquisito crescente interesse nello scenario industriale. I robot collaborativi, anche detti cobot, vengono sempre più dotati di sensori che permettono loro di percepire l'ambiente che li circonda. In particolare, per quanto riguarda la sincronizzazione delle operazioni svolte da uomo e robot, un ruolo fondamentale è occupato dai sensori di visione. Questi sensori consentono al robot di capire dove si trova l'uomo in un certo istante di tempo, di capire quale compito sta svolgendo e addirittura di prevedere le sue azioni future.

Alla base di queste capacità si trova l'algoritmo per la stima della posa dell'uomo. Stimare la posa da un'immagine è un problema non facile per numerose ragioni: complessità della struttura del corpo umano, variabilità nel fisico, condizioni di illuminazione ambientale, possibili occlusioni nella scena, perdita delle informazioni dovuta all'osservazione della posa da proiezioni di immagini planari 2D, eccetera. Questo problema è stato affrontato con diverse tecniche, tuttavia quella sicuramente più utilizzata per stimare la posa dell'uomo è quella il filtro di Kalman. Questa tecnica presenta però alcuni punti deboli, in particolare risulta complesso tenere in considerazione le occlusioni nel processo di stima.

Questo lavoro vuole fornire una soluzione al problema della stima della posa in situazioni di occlusione parziale dell'operatore. L'algoritmo di stima qui proposto si basa sulla tecnica del particle filter che, come vedremo, consente di introdurre vincoli nel processo di stima che tengono conto delle possibili occlu-

sioni. L'algoritmo è stato implementato per utilizzare i dati di profondità forniti dal sensore Kinect V2. Infine i risultati ottenuti sono stati confrontati con quelli ottenuti applicando il filtro di Kalman.

Chapter 1

Introduction

1.1 Application scenario

Industrial robots are the solution that best fits the flexible automation paradigm. Their mechanical structure and their programmability make industrial robots capable of performing a wide variety of tasks. If up to the recent past the paradigm universally adopted for industrial robotics provided for the strict segregation of robots in protected environments, nowadays the potentiality of a human-robot collaboration has been revalued. Referring to the classical example of an assembly process there are some actions which are too complex to be performed by a robot and others which require high precision and repeatability not suitable for the human. In these situations, collaborative robotics provides a huge benefit.

In this context, a crucial role is played by the vision sensors, which give to the robot the sense of sight. In the last years the market of vision sensors has been revolutionized by Microsoft X-Box Kinect Sensor, a depth camera that is used in the gaming industry to capture motions of players using the technology of an RGB camera and infrared camera to recognize depth. Because of its low-cost depth mapping sensor, Kinect has been largely implemented in all sorts of applications like unmanned aerial vehicle, human tracking, 3D model reconstruction,

robot navigation and medical implementations. Among all these applications, human tracking is particularly relevant for human-robot interaction. Defining the position of the human in space is indeed the starting point for safe interaction trajectory generation algorithms. Despite many years of research, however, pose estimation remains a very difficult and still largely unsolved problem. The most significant challenges are: variability of human visual appearance in images, variability in lighting conditions, variability in human physique, partial occlusions due to self articulation and layering of objects in the scene, complexity of human skeletal structure, high dimensionality of the pose, and the loss of 3D information that results from observing the pose from 2D planar image projections [2].

A problem which gains relevance in the human-robot collaboration scenario is the management of occlusion. Occlusions introduce uncertainty in the process of pose estimation and this is reflected in a limitation on the generation of trajectories that the robot has to follow. If occlusions were not taken into account, unsafe trajectories could be generated. On the other hand, overestimating the uncertainty could lead to an excessive limitation on the set of possible trajectories. Ultimately we can say that handling occlusion improves both safety and productivity.

1.2 Goal

At the state of the art, the most used technique to estimate the pose of the human body is for sure the Kalman filter. This technique is particularly simple and efficient from the computational point of view, however, it is quite difficult to take into account occlusions in its formulation. In literature there exist alternative techniques in order to estimate the pose, which allow to easily introduce arbitrary constraints in the estimation process. Among these techniques, there are the non-parametric filters such as the histogram filter and the particle filter which are characterized by representing the posterior distribution with samples of the state.

This sample-based nature makes the non-parametric filters the most versatile filtering techniques.

The goal of this work is to address the problem of human pose estimation in presence of occlusions. To do so we will make use of one of the aforementioned technique: the particle filter. Although this technique has been already used to tackle the problem of human pose estimation, in this work we present a method to consider also occlusions in the estimation process. Other works address the problems introduced by occlusions, however, these works start from some simplifying hypothesis such as occlusions a priori known or human's movements limited to a finite set [3]. Here we want to provide a more general and robust approach removing these assumptions in order to solve different scenarios at once.

1.3 Contributions

The main contribution given in this work is the novel approach to the human pose estimation robust to the case of occlusions. In particular, when occlusions occur, the method designed is able to limit the uncertainty relative to the pose in areas of space consistent with the shape of the occlusions and consistent with the constraints imposed by the anatomic distances. The method is based on the particle filter technique and uses the depth data provided by a depth sensor (Microsoft Kinect sensor V2). The proposed method has been applied to the human-robot collaborative scenario and its benefits in terms of safety and productivity have been proven.

1.4 Structure of the thesis

The thesis is organized as follow:

- **Chapter 2**: in this chapter, the state of the art regarding the pose estimation is presented. In particular, the concept of pose of the human body

is introduced and two ways to describe it are presented. Next, the Kalman filter technique is formulated to solve the problem of pose estimation. Then we describe the concept of occlusion in the computer vision framework and the behaviour of the Kalman filter when an occlusion occurs will be analysed. Alternative techniques which are better suited to include occlusions in their formulation are shown. Among them, the particle filter is particularly deepened. The chapter ends with an overview of how the pose estimation in presence of occlusion has been treated in literature.

- **Chapter 3**: here the instrumentation used is described. More in depth we describe the Microsoft Kinect sensor and YuMi, the collaborative robot used for the final demo.

- **Chapter 4**: this chapter describes the implementation of the algorithm proposed to estimate the human pose in presence of occlusions. First, we modeled the occlusion, starting from the depth map, as a constraint to impose in the particle filter formulation. Then some technical details regarding the implementation of the algorithm are discussed, such as the imposition of distance constraints or the estimation of process noise and measurement noise covariance matrices. Finally, in the last section of the chapter, a summary of the algorithm's operation is provided.

- **Chapter 5**: in this chapter the results obtained are presented. First, the performance of the particle filter is analysed. Then the effectiveness of the algorithm is shown through experimental results using the data provided by Kinect and finally, the algorithm is applied to a real collaboration situation and the results are presented.

Chapter 2

State of the Art

2.1 Introduction

In this chapter, we will introduce the concept of pose of the human body and we will show the main techniques used to estimate it. More in depth, in Section 2.3 two ways to describe the human pose are presented and then the mainly used technique for pose estimation is described: the Kalman filter. Afterwards, the problem introduced by occlusions in the process of estimation is treated and some techniques to cope with this problem are proposed. Among these techniques, the particle filter is studied in depth. Hence, a simple example of estimation in presence of occlusion is made. Finally, an overview of the use of particle filter to approach the human pose estimation is presented in Section 2.7.

2.2 Preliminaries

The pose of an object identifies the attributes of position and orientation of an object in space (Figure 2.1). The position defines the location of the object in a coordinate system while the orientation identifies the angles formed by the major axes of the object relative to a reference frame.

In computer vision, the task of estimating the pose of an object from an image is called pose estimation. By human pose, we refer to the configuration of the human body in space. The human pose can be described in different ways: the most common representation is obtained by describing the body with a kinematic model. For our purposes, we will focus only on the human upper body without loss of generality. The complete kinematic model of the full upper body motion is reported in the next section.

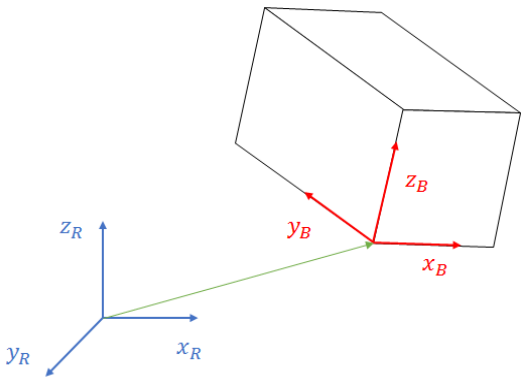


Figure 2.1: *Position and orientation of an object in space*

2.3 Kinematic model of the human motion

The human pose can be described using a 12-components vector:

$$p = (x, y, \theta, \rho, \alpha^{right}, \alpha^{left})^t$$

where x, y, θ represent the unicycle joint variables while $\rho, \alpha^{right}, \alpha^{left}$ are the torso flexion angle and the arms joint variables respectively, see Figure 2.2 . We make use of the human arm kinematics defined in [4].

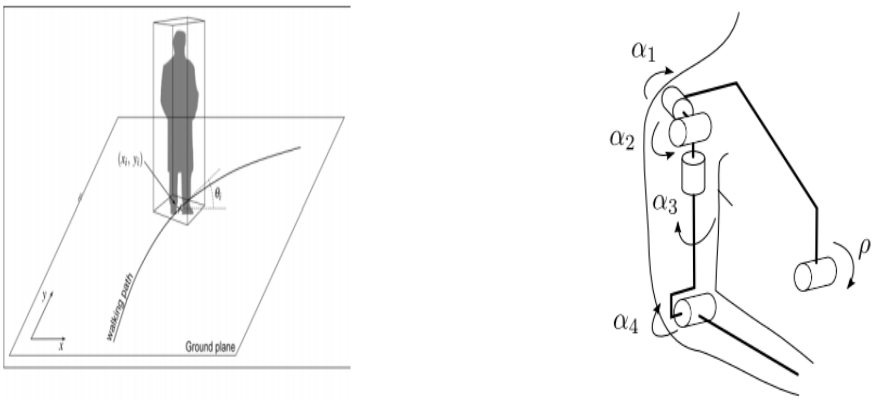


Figure 2.2: *Model of the human walking kinematic (right) and kinematic model of human arm and torso flexion angle ρ (left)*

The full motion of the upper human body results in a 3-dof base moving on the ground plane, one lumped 1-dof (flexion/extension) torso, a head (fixed) and two 4-dof arms. A graphical representation of such kinematic model is given in Figure 2.3.

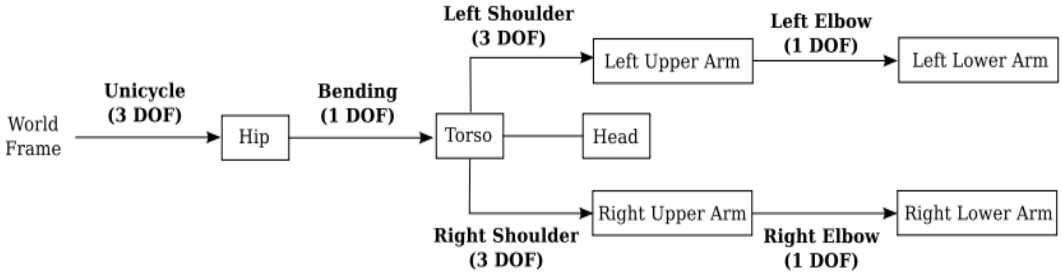


Figure 2.3: *Complete human kinematic model: DOFs, frames and bodies*

An analogous representation of this kinematic model can be given in terms of a set of 3D points composing a rough scheme of the human skeleton (Figure 2.4). Since only the motion of the upper part of the human body is considered, the points of interest are: thorax (T), neck (N), head (H), left shoulder (LS), right shoulder (RS), left elbow (LE), right elbow (RE), left wrist (LW), and right wrist (RW).

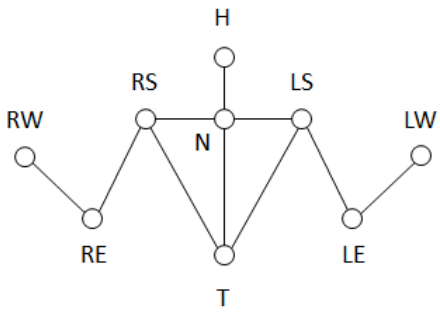


Figure 2.4: *Approximated human skeleton*

The skeletal representation is particularly useful because there are several algorithms in the literature (for instance [5]) that can be used to extract the skeletal points from a depth image which can be obtained using a depth sensor. In order to move from the skeletal representation to the corresponding kinematic one, an inverse kinematic procedure have to be implemented. Conversely, the opposite transformation requires a forward kinematic procedure. Details regarding forward and inverse kinematics are provided in Appendix A and Appendix B of[4].

2.4 Pose estimation with Kalman filter

In this section the most common technique for pose estimation is presented.
The Kalman filter (KF) is a widely used technique for state estimation. It assumes all the variables involved to follow a Gaussian distribution so that the posteriori distribution can be described in a modal way by its first and second order moments i.e. its mean μ and its covariance Σ. In case of linear systems, Gaussianity of variables is kept all the time and KF provides the optimal solution.

Consider a system in the following form:

$$x_k = A_k x_{k-1} + B_k u_k + w_k \tag{2.1}$$

$$y_k = C_k x_k + v_k \tag{2.2}$$

where x_k, u_k, y_k are state, input and output respectively, $w_k \sim N(0, Q_k)$ and $v_k \sim N(0, R_k)$ are the process and measurement noise.

The working principle of the KF algorithm is reported in Algorithm 1.

Algorithm 1 Closed loop Kalman filter

1: **procedure** KALMAN FILTER($\mu_{k-1}, \Sigma_{k-1}, u_k, y_k$)
2: $\overline{\mu}_k = A_k \mu_{k-1} + B_k u_k$
3: $\overline{\Sigma}_k = A_k \Sigma_{k-1} A_k^t + Q_k$
4: $K_k = \overline{\Sigma}_k C_k^t (C_k \overline{\Sigma}_k C_k^t + R_k)^{-1}$
5: $\mu_k = \overline{\mu}_k + K_k (y_k - C_k \overline{\mu}_k)$
6: $\Sigma_k = (I - K_k C_k) \overline{\Sigma}_k$
7: **return** μ_k, Σ_k
8: **end procedure**

At time k the inputs of the KF are the estimated values of μ and Σ at time $k-1$, the input u_k and the measurement y_k. The output are the mean μ_k which is the state estimate (also denote with $\hat{x}_k$) and the covariance matrix Σ_k. In order to provide these values the prior $\overline{\mu}_k$ and $\overline{\Sigma}_k$ are computed based on the input value u_k (lines 2-3). $\overline{\mu}_k$ (also denote with $\overline{x}_k$) and $\overline{\Sigma}_k$ represent the predicted state estimate. The prior belief is subsequently transformed in the desired belief by incorporating the measurement y_k (lines 4 to 6). The variable K_k computed in line 4 is the so called *Kalman gain* and represents the degree according to which the measurement is incorporated in the new estimate. Figure 2.5 shows the working principle of the KF technique.

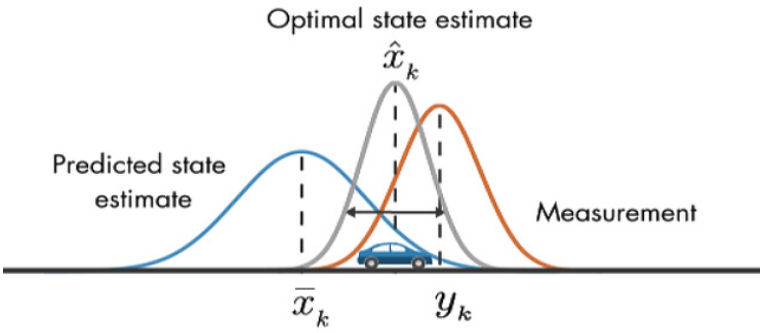


Figure 2.5: *Working principle of KF in a positioing application*

For the sake of simplicity let's assume that no input u_k is available: at each time instant k the algorithm provides an updated value of the mean μ_k and of the covariance Σ_k based on the previous ones and on the incoming measurement data y_k.

If no measurements are available the KF algorithm runs in open loop (Algorithm 2). This behavior will be analysed in the next section while dealing with occlusions (Section 2.5).

Algorithm 2 Open loop Kalman filter

1: **procedure** OPEN LOOP KALMAN FILTER(μ_{k-1}, Σ_{k-1})
2: $\mu_k = A_k \mu_{k-1}$
3: $\Sigma_k = A_k \Sigma_{k-1} A_k^t + Q_k$
4: **return** μ_k, Σ_k
5: **end procedure**

The pose estimation problem can be reformulated in terms of state estimation. Recalling that $p = (x, y, \theta, \rho, \alpha^{right}, \alpha^{left})^t$, consider the problem of estimating the pose by joint variables. The adopted process model consist in a chain of three discrete time integrators for each joint variable. In detail:

$$s_{k+1} = As_k + \eta_k \tag{2.3}$$

$$s_k = \begin{pmatrix} p_k \\ \dot{p}_k \\ \ddot{p}_k \\ \dddot{p}_k \end{pmatrix} \quad A = \begin{pmatrix} I & \Delta tI & \frac{\Delta t^2}{2}I & \frac{\Delta t^3}{6}I \\ O & I & \Delta tI & \frac{\Delta t^2}{2}I \\ O & O & I & \Delta tI \\ O & O & O & I \end{pmatrix}$$

$\eta_k \sim N(0, G)$ where G is a diagonal matrix parametrized as follows: for each block of the state vector we consider the corresponding first truncated element of the Taylor expansion contained in matrix A:

$$G = \begin{pmatrix} \sigma_p^2 \frac{\Delta t^4}{24} I & O & O & O \\ O & \sigma_{\dot{p}}^2 \frac{\Delta t^3}{6} I & O & O \\ O & O & \sigma_{\ddot{p}}^2 \frac{\Delta t^2}{2} I & O \\ O & O & O & \sigma_{\dddot{p}}^2 \Delta t I \end{pmatrix}$$

where standard deviations σ_p, $\sigma_{\dot{p}}$, $\sigma_{\ddot{p}}$, $\sigma_{\dddot{p}}$ are tunable parameters.

Concerning the output equation, since the inverse kinematics from skeletal points to joint variables can be computed in closed form, we consider as output vector the measure of all joint positions so that we have:

$$y_k = Cs_k + \zeta_k \tag{2.4}$$

$$C = \begin{pmatrix} I_{12\times 12} & O_{12\times 36} \end{pmatrix}$$

$\zeta_k \sim N(0, R)$ represents the output noise and $R = \sigma_y^2 I$ is the covariance matrix. On the system described by the equations (2.4) and (2.5) it is possible to apply the KF and retrieve the state estimate.

2.5 Kalman filter behaviour under occlusions

The aim of this section is to introduce the concept of occlusion and show the issues that arise using a Gaussian filter like the KF when occlusions occur.

First we define what an occlusion is. Considering the object tracking framework an occlusion is a situation in which one or multiple objects partially or completely hide the target object. Let's analyse a simple case of position estimation of an object moving through occlusions in the xy plane. The system can be described by the following equations:

$$s_k = A_k s_{k-1} + \eta_k \tag{2.5}$$

$$z_k = C_k s_k + \zeta_k \tag{2.6}$$

where s_k represent the state, z_k the measurement vector, η_k is the process noise and ζ_k is the measurement noise. Let's suppose that we want to estimate the state $s_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^t$ where the first two components are the x and y coordinates of the object center of mass and $\dot{x}_k$ and $\dot{y}_k$ are the speed in the x and y direction respectively. The measurement vector z_k is formed by the x and y position of the object measured by a vision sensor. In the following the dynamic matrix and the observation matrix are reported:

$$A = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$\eta_k \sim N(0, G)$ models the process noise; its covariance G can be parametrized as follows:

$$G = \begin{pmatrix} \sigma_p^2 \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & \sigma_p^2 \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 0 & \sigma_{\dot{p}}^2 \Delta t & 0 \\ 0 & 0 & 0 & \sigma_{\dot{p}}^2 \Delta t \end{pmatrix}$$

$\zeta_k \sim N(0, R)$ is the measurement noise and its covariance matrix R can be modeled like $R = \sigma_y^2 I$.

Suppose now that the object is moving in the xy plane and no occlusions occur. The situation will appear similar to the one in Figure 2.6:

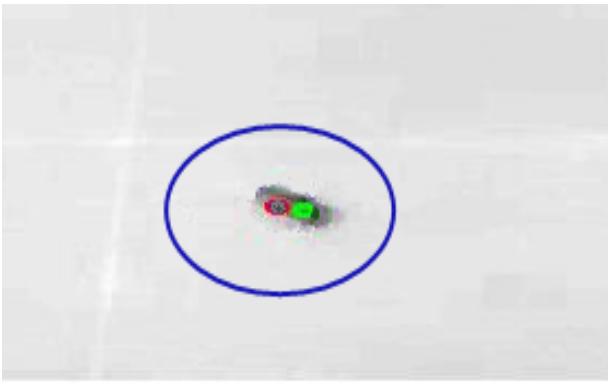


Figure 2.6: *Object tracking with KF*

On the object three different circles are drawn: the green one is centered in the x and y coordinates given by the vision sensor, the red one defines the first two components of the estimated state vector obtained using the KF and the blue circle is the uncertainty relative to the position estimate fixed a certain confidence level. The ellipse can be computed as $e_x^t \Sigma^{-1} e_x = \chi_\alpha^2$ where e_x is the state error vector and α is the confidence level considered.

Let's now consider the case when the object to be tracked is occluded (see Figure 2.7).

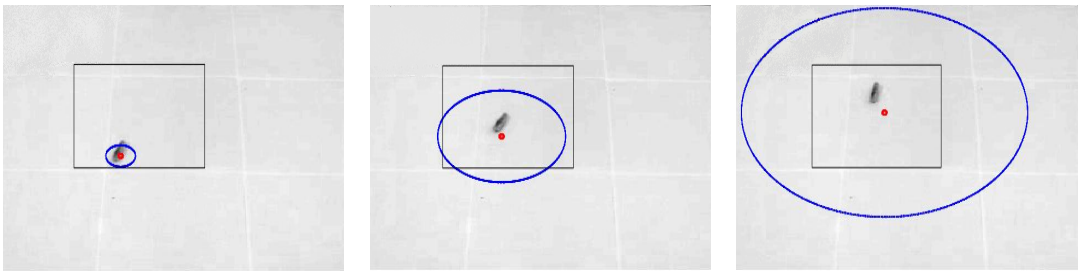


Figure 2.7: *KF behaviour under occlusion*

The occlusion is represented by the black rectangle. As we can see from the picture when the object is occluded, the green circle disappears because the vision sensor cannot provide a valid position measurement and the estimation proceeds in open-loop (see Algorithm 2). The position is computed by only considering the dynamics of the system and the uncertainty starts growing until new measurement data are available. Note that, from a certain time instant on, the ellipse associated with the position uncertainty becomes bigger than the occlusion itself so that we have non zero probability to find the object outside the occlusion. This situation is clearly paradoxical because if the object was really outside the occlusion the vision sensor could provide a position measurement. This inconsistency comes from the way KF represents the estimate: Gaussian description of the uncertainty is not coherent with the shape of the occlusion. While the occlusion occurs our aim is to have a probability distribution of the estimate consistent with the constraints imposed by the occlusion itself. In the next section, a set of constrained Bayesian estimators will be presented and it will be shown how they can be adapted to solve the problem of pose estimation in presence of occlusions.

2.6 Constrained Bayesian estimators

Before entering the analysis of the techniques considered, we invite the reader to read the Appendix A related to the Bayes' estimation problem.

2.6.1 Unscented Kalman Filter

The first technique examined is the unscented Kalman filter (UKF). This technique belongs to a bigger class of filters called sigma-point Kalman filters or linear regression Kalman filters, which use the statistical linearization technique [6], [7]. This technique is used to linearize a nonlinear function of a random variable through a linear regression between a set of points drawn from the prior distribution of the random variable. The UKF is founded on the intuition that *it is easier to approximate a probability distribution than an arbitrary nonlinear function* [8]. The posterior distribution is directly approximated by a set of deterministic points called sigma points.

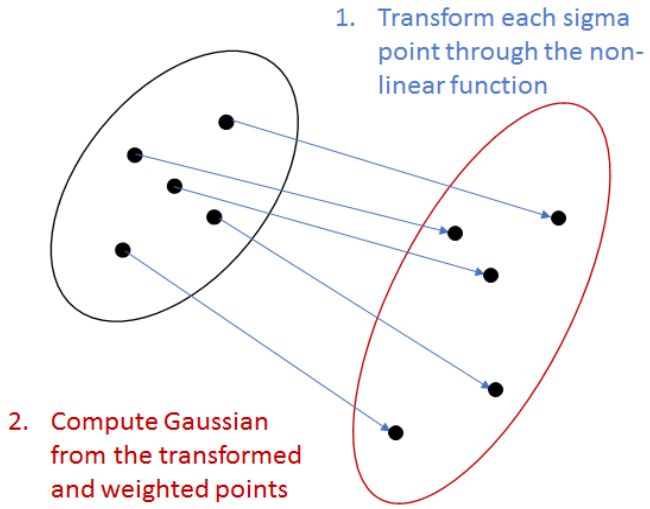


Figure 2.8: *Unscented Kalman filter: two steps working principle*

Several algorithms have been developed during the years, but in this work the symmetric sigma points formulation is presented . Refer to Algorithm 3.

Algorithm 3 Unscented Kalman filter

procedure UNSCENTED KALMAN FILTER($\hat{x}_{k-1}, P_{x_{k-1}}, u_k, y_k$)
  for i=1 to 2n **do**
    if 1≤ i ≤ n **then**
      $\tilde{x}^i = [row_i(\sqrt{nP_{x_{k-1}}})]^t$
    else if n+1≤ i ≤ 2n **then**
      $\tilde{x}^i = [-row_i(\sqrt{nP_{x_{k-1}}})]^t$
    end if
    $X_{k-1} = \{\hat{x}_{k-1} + \tilde{x}^i\}$
    $X^-_{k,i} = f_k(X_{k-1,i}, u_k)$
  end for
  $\hat{x}^-_k = \sum_{i=1}^{2n} W^x_i X^-_{k,i}$
  $P^-_{x_k} = \sum_{i=1}^{2n} W^c_i (X^-_{k,i} - \hat{x}^-_k)(X^-_{k,i} - \hat{x}^-_k)^t + Q_k$
  for i=1 to 2n **do**
    $\gamma_{k,i} = h(X^-_{k,i})$
  end for
  $\hat{y}_k = \sum_{i=1}^{2n} W^x_i \gamma_{k,i}$
  $P_{y_k y_k} = \sum_{i=1}^{2n} W^c_i (\gamma_{k,i} - \hat{y}_k)(\gamma_{k,i} - \hat{y}_k)^t + R_k$
  $P_{x_k y_k} = \sum_{i=1}^{2n} W^c_i (X^-_{k,i} - \hat{x}^-_k)(\gamma_{k,i} - \hat{y}_k)^t$
  $K_k = P_{x_k y_k} P^{-1}_{y_k y_k}$
  $\hat{x}_k = \hat{x}^-_k + K_k(y_k - \hat{y}_k)$
  $P_{x_k} = P^-_{x_k} - K_k P_{y_k y_k} K^t_k$
  return $\hat{x}_k, P_k$
end procedure

W^x_i and W^c_i are the state weight and covariance weight respectively. Also UKF

algorithm follows a prediction/correction structure. The sigma points are processed through the nonlinear model of the system, producing a set of propagated sigma points. By choosing appropriate weights, the Kalman gain is computed and the prediction estimate is corrected according to the measurement evidence.

The description of the probability distribution through a set of sigma points introduces the possibility to impose constraints in the process of state estimation. This strategy is known in literature as clipping technique [9]. The basic idea is to set up a minimization problem in this form:

$$\begin{aligned} \min_{\tilde{x}_k} \quad & (\tilde{x}_k - \hat{x}_k)^t W_k (\tilde{x}_k - \hat{x}_k) \\ \text{subject to} \quad & D_k \tilde{x}_k \leq d_k \end{aligned} \tag{2.7}$$

where $\tilde{x}_k$ denotes the constrained state. equation (2.7) can be rewritten as a QP problem like:

$$\begin{aligned} \min_{\tilde{x}_k} \quad & \tilde{x}_k^t W_k \tilde{x}_k - 2\hat{x}_k^t W_k \tilde{x}_k \\ \text{subject to} \quad & D_k \tilde{x}_k \leq d_k \end{aligned} \tag{2.8}$$

Note that if $W_k = I$ and $D_k = I$ the solution to the minimization problem is $\tilde{x}_k^t = d_k$ when $\hat{x}_k$ violates the constraints. The application of constraints can be done in different phases of the algorithm and a reformulation in terms of maximum likelihood can be given:

$$\min_{\tilde{x}_k^-} \quad (\tilde{x}_k^- - \hat{x}_k^-)^t (P_{x_k}^-)^{-1} (\tilde{x}_k^- - \hat{x}_k^-) \qquad \text{subject to} \qquad D_{x_k^-} \tilde{x}_k^- \leq d_{\tilde{x}_k^-} \tag{2.9}$$

$$\min_{\tilde{y}_k} \quad (\tilde{y}_k - \hat{y}_k)^t P_{y_k y_k}^{-1} (\tilde{y}_k - \hat{y}_k) \qquad \text{subject to} \qquad D_{\tilde{y}_k} \tilde{y}_k \leq d_{\tilde{y}_k} \tag{2.10}$$

$$\min_{\tilde{x}_k} \quad (\tilde{x}_k - \hat{x}_k)^t P_{x_k}^{-1} (\tilde{x}_k - \hat{x}_k) \qquad \text{subject to} \qquad D_{\tilde{x}_k} \tilde{x}_k \leq d_{\tilde{x}_k} \tag{2.11}$$

Although this technique can be used for our purposes, in the literature there are estimation techniques that dominate the UKF in terms of estimation error and computational effort [10].

A non-parametric filtering technique will now be presented. The next technique tries to approximate directly the posterior distribution by a set of finite samples. We are talking about the particle filter.

2.6.2 Particle filter

Particle filter (PF) is an estimation technique which does not rely on linearization techniques or Gaussianity assumption. It tries to approximate the posterior distribution by a set of finite samples called particles. This approximation can represent a much broader space of distribution than the unimodal Gaussian one [11].

We denote the set of particles at time instant k by $X_k = \{x_k^{(1)}, \cdots, x_k^{(N)}\}$ where N is the total number of particles in the set. Each particle $x_k^{(i)}$ represents an instantiation of the state at time k i.e. an hypothesis on the true value of the state. The posteriori distribution can be approximated by a discrete sum as follows:

$$\hat{p}(x_k|y_{1:k}) = \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i) \tag{2.12}$$

where w_k^i is the weight associated to the particle x_k^i and $\delta(\cdot)$ is the Dirac delta function.

The PF algorithm can be subdivided into three phases:

1. *Motion update*: starting from a set of particles X_{k-1} a new set $\overline{X}_k$ is obtained where each x_k^i is drawn proportional to a proposal distribution $q(x_k|y_{1:k})$. A common choice for the proposal distribution is $p(x_k|u_k, x_{k-1})$.

2. *Measurement update*: in this phase, for each particle, the algorithm computes the probability that, given the state of the particle, the output is the one actually measured. It assigns a weight w_k^i for each particle proportional to the said probability.

3. *Resampling*: a new set of N particles X_k is drawn from the previous belief with probability proportional to the weight w_k^i. Particles consistent with sensor readings are more likely to be chosen (possibly more than once) and particles inconsistent with sensor readings are rarely picked. Denser a subregion of the state space is populated by samples, more likely it is that the true state falls into this region.

Concerning the resampling phase, this is particularly important in order to avoid degeneration problems [12]. The resampling step has the important function to force particles back to the posterior. In literature, there are different kind of resampling techniques like the Multinomial Resampling (MR), the Stratified Resampling (SR), the Residual Resampling (RR), etc... [13]. In this work, we will use the Bootstrap Resampling (BR). The BR algorithm is characterized by the following steps:

1. the weight associated to each state particle is normalized;
2. computation of an array containing the cumulative sum of the normalized weights;
3. a random number is drawn from a uniform distribution 0-1;
4. the first index relative to the element in the array of cumulative sum which has a value greater o equal to the random number generated before is considered (Figure 2.9);
5. the particle corresponding to the index selected is replicated.

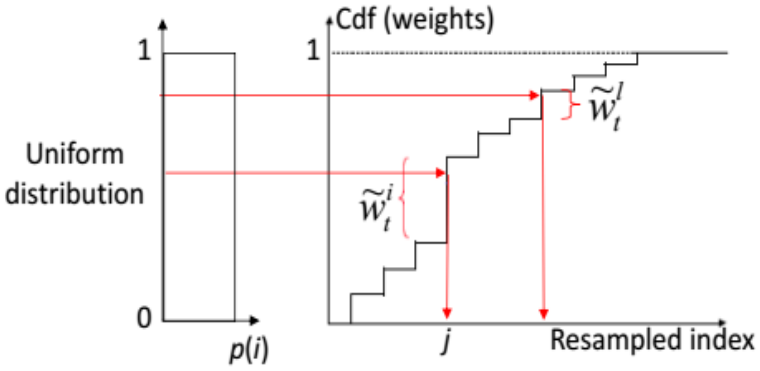


Figure 2.9: *Bootstrap Resampling procedure taken from [1])*

Algorithm 4 describes the PF algorithm taken from [11]:

Algorithm 4 Closed loop particle filter

1: **procedure** PARTICLE FILTER(X_{k-1}, u_k, y_k)
2: $\overline{X}_k = X_k = \emptyset$
3: **for** i=1 to N **do**
4: sample $x_k^i \sim p(x_k|u_k, x_{k-1})$
5: $w_k^i = p(y_k|x_k^i)$
6: $\overline{X}_k = \overline{X}_k + x_k^i, w_k^i$
7: **end for**
8: **for** i=1 to N **do**
9: draw x_k^i with probability $\sim w_k^i$
10: add x_k^i to X_k
11: **end for**
12: **return** X_k
13: **end procedure**

In the first for loop the particles are generated according to the proposal distribution and they are weighted proportionally to $p(y_k|x_k^i)$, while in the second loop the resampling phase is performed.

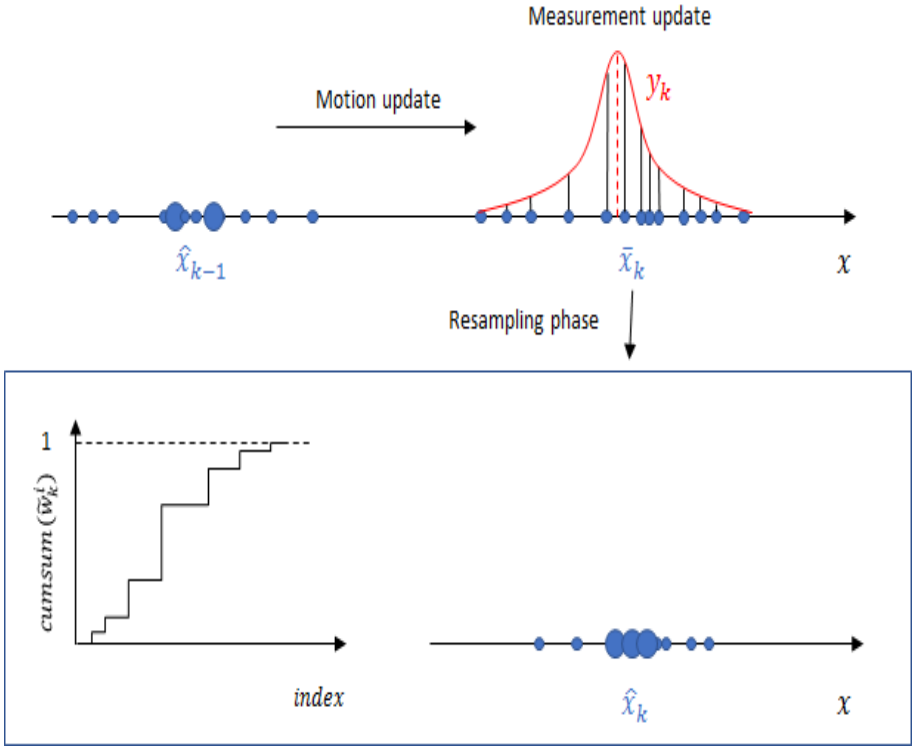


Figure 2.10: *Phases of the PF algorithm*

The open loop behaviour is also reported in Algorithm 5.

Algorithm 5 Open loop particle filter

1: **procedure** OPEN LOOP PARTICLE FILTER(X_{k-1}, u_k)
2: $\overline{X}_k = X_k = \emptyset$
3: **for** i=1 to N **do**
4: sample $x_k^i \sim p(x_k|u_k, x_{k-1})$
5: $w_k^i = 1/N$
6: $\overline{X}_k = \overline{X}_k + x_k^i, w_k^i$
7: **end for**
8: **return** X_k
9: **end procedure**

Note that when no measurements are available all the particles have the same

weight equal to$1/N$. In fact there is no reason to weight more a particle with respect to another one.

The sample-based nature of the PF algorithm facilitates the process of including constraints in the state estimation problem. An effective way to perform this is to define a likelihood function [9]:

$$L_c(x_k^i) = \begin{cases} 1, & \text{if } x_k^i \in \mathbb{X}_k \\ 0, & \text{if } x_k^i \notin \mathbb{X}_k \end{cases} \tag{2.13}$$

where $\mathbb{X}_k$ represent a certain state constraint region at time k. The idea is to assign $w_k^i = L_c(x_k^i)p(y_k|x_k^i)$. This modification enables the algorithm to discard all the particles violating constraints. The advantage of acceptance/rejection scheme is twofold. First, it guarantees the particles to stay in constraint region and nearly no extra computation cost is needed. Second, the method retains the Monte Carlo sampling feature of PF which makes it suitable for non-Gaussian problems.

Let's analyse the example introduced in the previous Section 2.5 applying now the constrained PF.

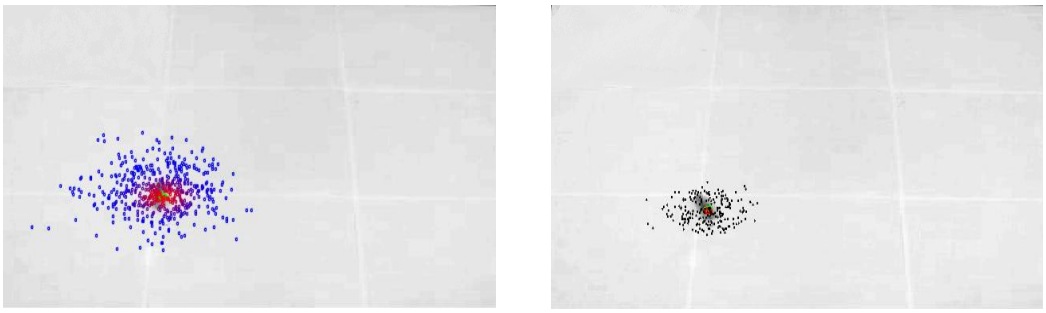


Figure 2.11: *On the left particles before the resampling phase; on the right particles after the resampling phase*

Figure 2.11 shows a non-occluded situation. In the picture on the left we can see the particles coming from the motion update phase and their relative weights computed in the measurement update phase: particles take a color between blue,

which corresponds to a weight equal to zero, and red associated to the maximum weight. Note that each particle has been weighted according to:

$$w_k^i = \frac{exp(-\frac{1}{2}(y_k - y_k^i)^t R^{-1}(y_k - y_k^i))}{\sqrt{|2\pi R|}} \tag{2.14}$$

where in our example $y_k^i = CAs_{k-1}^i$. On the right we can see in black the particles which survive after the resampling phase, in red the position estimate (the "center of mass" of all particles) and the measurement data in green.

Assuming that the shape and position of the occlusion is known to us at each time instant k let's analyse what happens to the estimation process in the occluded scenario (Figure 2.12).

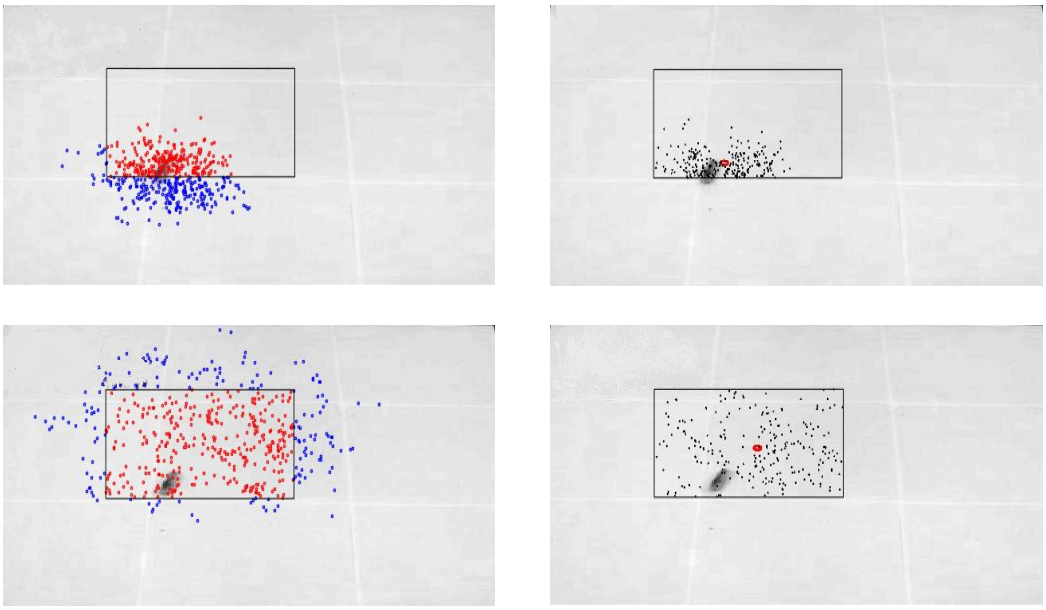


Figure 2.12: *Constrained PF behaviour under occlusion*

When the object goes under the occlusion we lose the measurement from the vision sensor and the uncertainty starts to grow but it remains limited inside the occlusion. Note that all the particles outside the rectangle have a weight equal to zero because the function $L_c(x_k^i)$ is zero for all that particles while the inside ones are all equally weighted. When the object goes under occlusion the particles

start propagating in open loop. After a sufficient time interval the probability to find the object in any position under the occlusion becomes uniform.

2.7 PF for pose estimation in literature

Particle filter has already been used to approach the problem of pose estimation and in some case also to treat the occlusion issues. There are several examples in literature. Here we want to name just a few to offer a complete overview of how this problem has been treated.

[14] addresses the full-body articulated human motion tracking from multi-view video sequences. The tracking is formulated as a multi-dimensional non-linear optimisation and solved using particle swarm optimisation (PSO), a PF-derived algorithm which has gained popularity in recent years due to its ability to solve difficult non-linear optimisation problems. The joints in the kinematic tree are optimised in a sequence, starting with the torso and proceeding towards the arms. This follows the inherent hierarchical structure of the human body, where the configuration of the joints at higher levels of the kinematic tree constrains that of joints appearing at lower levels.

The hierarchical approach to estimate the pose has been used also in [15]. In this work, the authors tackle the hand tracking problem with a model-based approach. The hand is tracked using the Hierarchical Model Fusion framework (HMF), first proposed by Makris et al. [16], which is a particle filter (PF) variant that decomposes the initial problem into smaller and simpler problems and efficiently addresses the implications of the high dimensionality (a problem which has been encountered also in our case study).

In [3] an attempt to cope with occlusion is made. This work presents an improved motion model based on the intuition that people tend to follow efficient trajectories through their environments rather than random paths. The proposed motion model learns common destinations within the environment by clustering

training examples of trajectories, then uses a path planner to predict how a person would move along routes from his or her present location to these destinations. The motion model was integrated into a particle-filter-based person-tracker, and it was experimentally demonstrated that the new motion model performs significantly better than simpler models, especially in situations in which there are extended periods of occlusion during tracking.

Finally in the work [17] the idea emerges that PF can also use different kinds of additional information that may be available, with the use of likelihood functions and sampling distributions. Such information may arise from targets having constraints in their motion. The ability to incorporate such kinematic behavior into the tracking algorithm can improve tracking performance. Kyriakides introduces the constrained motion proposal (COMP) algorithm that uses multi-target proposal densities and motion models that incorporate kinematic constraint information into a particle filter. More specifically, it uses methods of sampling and likelihood functions that take into account motion constraint information. This idea will be exploited in the following during the implementation.

Chapter 3

Instrumentation

3.1 Introduction

In the following, the instrumentation used for the algorithm implementation will be presented. First, we will describe the Microsoft's Kinect sensor used to retrieve the skeletal points and the depth data regarding the environment. Then we will give a brief description of the robot YuMi, used in the final demo.

3.2 Microsoft Kinect

A recent development in range sensing technology is Microsoft's Kinect sensor (Microsoft, 2010). Kinect was primarily designed for natural interaction in a computer game environment. However, the characteristics of the data captured by Kinect have attracted the attention of researchers from the field of mapping and 3D modeling [18].

On February 21-2011, Microsoft announced that it would release a non-commercial Kinect software development kit (SDK) for Microsoft Windows in spring 2011, and the first beta was released for Windows 7 on June 16, 2011. The SDK includes Windows 7 compatible PC drivers for Kinect device. It pro-

vides Kinect capabilities to developers to build applications with C++, C#, or Visual Basic by using Microsoft Visual Studio 2010 and includes the following features:

- raw sensor streams: access to low-level streams from the depth sensor, color camera sensor, and four-element microphone array;
- skeletal tracking: the capability to track the skeleton image of one or two people moving within Kinect's field of view for gesture-driven applications;
- advanced audio capabilities: audio processing capabilities include sophisticated acoustic noise suppression and echo cancellation, beam formation to identify the current sound source, and integration with Windows speech recognition API;
- sample code and documentation.

The Kinect contains three components that work together to detect human motion and create the physical image on the screen. These three components are:

1. RGB color VGA video camera: the camera detects the red, green, and blue color components as well as body-type and facial features. This helps in facial recognition and body recognition.

2. Depth sensor: the depth sensor contains a monochrome CMOS sensor and infrared projector that help create the 3D imagery measuring the disparity of the resulting image respect to a reference pattern (it will be better explained in the following).

3. Multi-array microphone: the microphone is an array of four microphones that can isolate the human voices from other background noises allowing human to use their voices as an added control feature.

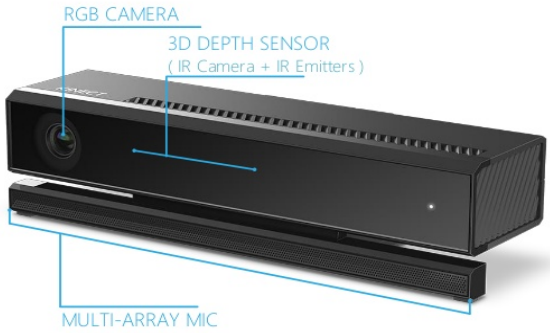


Figure 3.1: *Microsoft Kinect V2*

The Kinect used in this work is the Microsoft Kinect Version 2. The technical details of the depth sensor are reported in Table 3.1.

Table 3.1: *Microsoft Kinect V2 datasheet*

Color camera resolution	1920×1080
Depth camera resolution	512×424
Accuracy (0.5 m to 3 m)	2 mm
Frame rate	$30 fps$
Min. depth distance	50 cm
Max. depth distance	8 m
Depth horizontal FOV	70°
Depth vertical FOV	60°
Skeleton joints defined	25
Max. people tracked	6

The data have been retrieved from the Microsoft MSDN and the study conducted in [19].

3.3 YuMi

Figure 3.2: *YuMi ABB*

YuMi is a collaborative robot designed by ABB company. Officially launched in September 2014, YuMi is conceived to work side-by-side with humans in manufacturing environments. The soft padded dual arms and the collision detection feature ensure the complete safety of the robot's co-workers. Whereas most heavy-manufacturing robots are aimed at performing precise tasks cyclically, YuMi is designed to replicate a human assembly worker, making it extremely flexible. YuMi is characterized by a lightweight structure (38 Kg) and high accuracy (0.02 mm). In the recent years, YuMi has been used in operation of assembly, packaging, gluing, dispensing and palletizing.

The end effectors of YuMi are endowed with grippers and suctions and can even come equipped with integrated vision systems.

The programming task of the robot is also simplified: users can exploit a "lead-through" to reduce the weight of the arms and allow for manipulation. The position of the arms can be changed and the joint configuration can be saved to provide an outline of the desired path. The saved trajectory be tested and changed when necessary.

Chapter 4

Methods for pose estimation in case of occlusions

4.1 Introduction

In this chapter, we will focus on the implementation. First, we use the depth data provided by the Kinect sensor to model the occlusions as constraints to impose in the PF formulation. Then we choose the representation of the human pose comparing the advantages and disadvantages of operating in the Cartesian space rather than in the joint space. Next we define the cases of occlusion to consider and finally, we report a summary of the algorithm implemented.

4.2 Depth map

The basic principle behind the Kinect depth sensor is the emission of an IR pattern and the simultaneous image capture by an IR camera. The laser source emits a single beam which is split into multiple beams by diffraction to create a constant pattern of speckles projected onto the scene. This pattern is captured by the IR camera and is compared to a reference pattern [20]. The reference pattern

is obtained by capturing a plane at a known distance from the sensor and is stored in the sensor memory. When a speckle is projected on an object whose distance to the sensor is smaller or larger than the one of the reference plane, the position of the speckle in the infrared image will be shifted in the direction of the baseline between the laser projector and the perspective center of the infrared camera. These shifts are measured for all speckles by a simple image correlation procedure, which yields to a disparity image. For each pixel, the distance from the sensor can then be retrieved from the corresponding disparity (this will be better explained in the following).

Figure 4.1 illustrates the relation between the distance of a point object k from the sensor relative to a reference plane and the measured disparity d. To express the 3D coordinates of the point object we consider a depth coordinate system with:

- origin at the perspective center of the infrared camera;
- Z axis is orthogonal to the image plane and pointing towards the object;
- the X axis perpendicular to the Z axis in the direction of the baseline b between the infrared camera center and the laser projector;
- Y axis orthogonal to X and Z forming a right handed coordinate system.

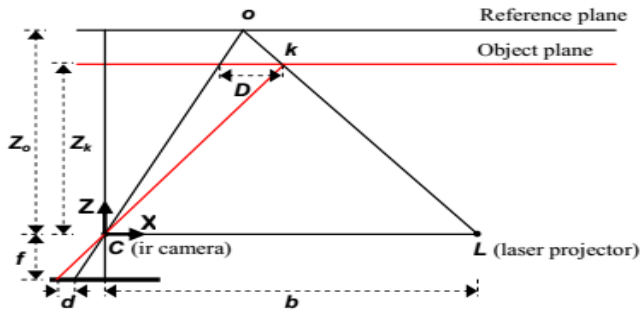


Figure 4.1: *Schema of the depth disparity relation*

Assume that the reference plane is at Z_0. If an object is moving from the reference plane closer or further away, the corresponding location of the speckle on the image plane will be displaced in the X direction (on the right or on the left respectively). The displacement can be measured in the image plane as a disparity d. From the similarity of triangles we can write:

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \tag{4.1}$$

$$\frac{d}{f} = \frac{D}{Z_k} \tag{4.2}$$

where Z_k denotes the depth of the point k in object space, b is the base length, f is the focal length of the infrared camera, D is the displacement of the point k in object space, and d is the observed disparity in image space. Rearranging the equations we can obtain:

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{fb}d} \tag{4.3}$$

Equation (4.3) is the basic formula to derive depth from the observed disparity, provided that the constant parameters Z_0, f and b can be determined by calibration. The X and Y components can be computed in the following way:

$$X_k = -\frac{Z_k}{f}(x_k - x_0 + \delta x) \tag{4.4}$$

$$Y_k = -\frac{Z_k}{f}(y_k - y_0 + \delta y) \tag{4.5}$$

where x_k and y_k are the image coordinates of the point, x_0 and y_0 are the coordinates of the principal point, and δx and δy are corrections for lens distortion, for which different models with different coefficients exist [20].

Microsoft Kinect can provide depth values in two different spaces: the camera space and the depth space. Camera space refers to the 3D coordinate system used by Kinect. The coordinate system is defined as in Figure 4.2 (same as the one previously described).

Figure 4.2: *Camera space coordinate system*

Depth space is the term used to describe a 2D location on the depth image. It can be described as a matrix of pixel where i is the row and j is the column. So $i = 0$, $j = 0$ corresponds to the top left corner of the image and $i = 423$, $j = 511$ is the bottom right corner of the image. At each pixel is associated a depth value, in mm, corresponding to the Z coordinate of the point in camera space. Depth data are expressed by default in the depth space.

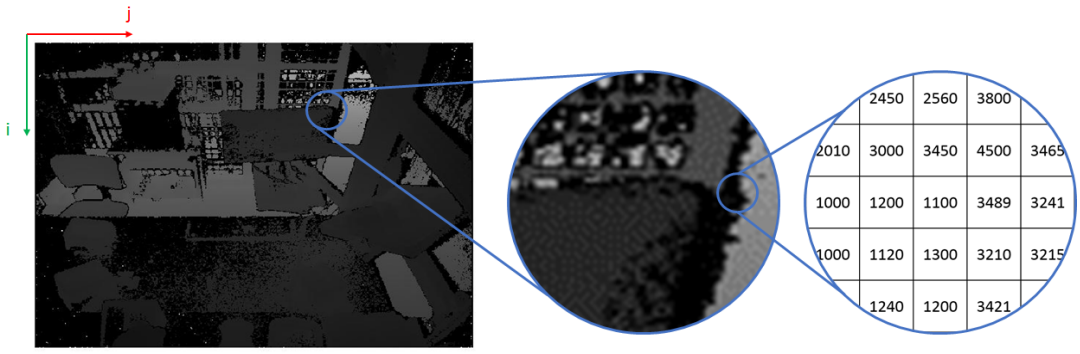


Figure 4.3: *Depth space representation*

4.3 Model of the occlusion

In Chapter 2 we showed an example of tracking an object moving in an environment where occlusions occur. In that situation, it was assumed that the geometric description of the occlusion was known. In this way, it was possible to exclude particles out of the occlusion. However, in our case of study, the only

information relative to the surrounding environment is provided by the depth information coming from Kinect.

Before entering into the analysis of the case of study, consider the problem of tracking a point object in the 3D space. Once again consider a state equation and output equation like in (4.6), (4.7), where the state is made for instance by the x, y, z object position coordinates and the corresponding velocity components.

$$s_k = A_k s_{k-1} + \eta_k \tag{4.6}$$

$$y_k = C_k x_k + \zeta_k \tag{4.7}$$

Suppose that the available measurements come from an object detection procedure implemented in the Kinect sensor which provides the x, y and z coordinates of the camera space. Also assume that this procedure provides a confidence value with which the measurement is retrieved. If this confidence value is below a certain threshold the measurement is considered not valid.

The situation when the measured value is not valid can be essentially due to:

1. high noise on the measurement;

2. occlusions.

The occurrence of the first situation can be limited by trying to eliminate the sources of noise such as diffused ambient light. Occlusions are more difficult to avoid, especially if they are inherent in the application considered.

When an occlusion occurs another object hides our target interposing between the latter and the sensor. Consider a reference frame of the sensor like in Figure 4.2. If an object occludes our target it means that its z coordinate is less than the one where the target object is located (see Figure 4.4). On the contrary if the object is not in occlusion its z is less than the background behind it.

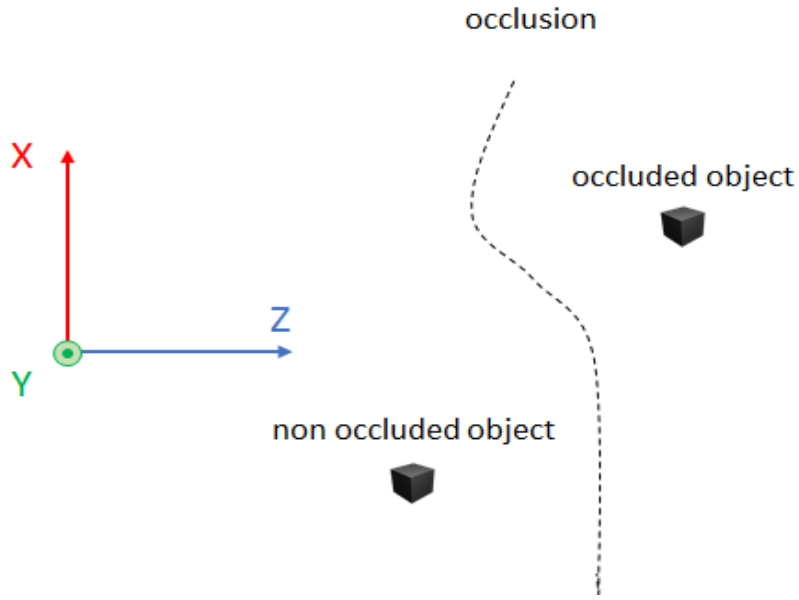


Figure 4.4: *Example of occlusion. The object is moving in the space at a fixed y coordinate*

Now our goal is to estimate the state of the system using the particle filter technique. How to impose constraints on the particles evolution? To answer this question consider the following situation:

- at time k the object is visible and a valid measurement is given by the sensor. The particle filter provides the estimate of position and velocity of the object at that time instant: $\hat{p}_k$ and $\hat{v}_k$ respectively;
- at time $k + \Delta t$ the object is occluded and the sensor doesn't provide any measurement, so the particle filter runs in open loop (see Algorithm 5).

The situation is pictorially represented in Figure 4.5. Observe that at time $k+\Delta t$ the object is under occlusion, the particles are updated according to the state equation and the relative weight is equal for all particles. Among these particles the only surviving ones are those with a z coordinate which is greater than the z of points at the same x and y in the depth map. This leads to a propagation of the particles consistent with the shape of the occlusion.

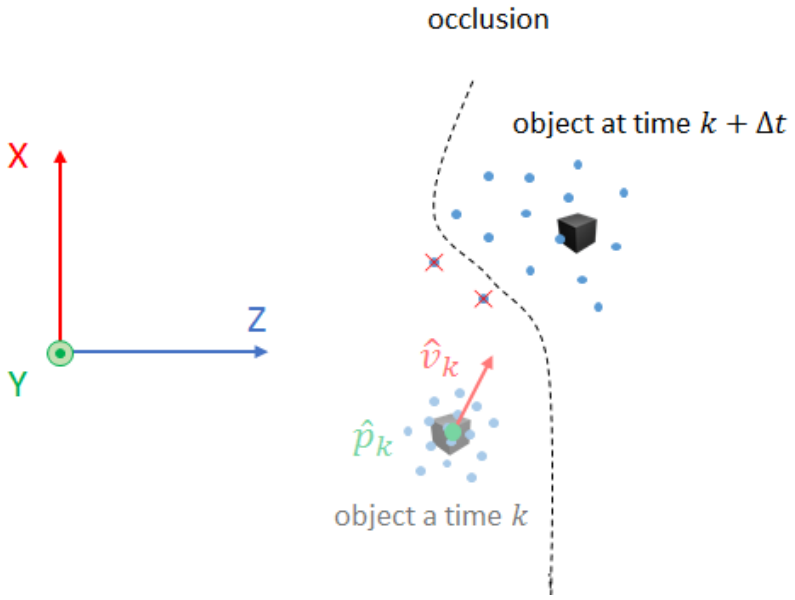


Figure 4.5: *Model of the occlusion as a constraint in the PF formulation*

In the previous statement, there was an implicit assumption: there exists a point in the depth map at the same x and y coordinate to those of the particle. In practice, this assumption doesn't hold. In fact, the probability that this situation happens is close to zero. Therefore we should compare the z of the point in z of the particle with the one of the point in the depth map which has the closest x and y to those of the particle. In order to do this, however, it is necessary to look at each point in the depth map and check its x and y coordinates. This is clearly computationally inefficient especially when the number of points in the depth map is very high (in our case considering a resolution of 512×424 the number of points is equal to 217088). The solution adopted to solve this problem is to map the coordinates of the particles in the depth space. Remember that in the depth space the image captured by the Kinect is described by a matrix M of pixels in which each element m_{ij} is a depth value expressed in millimeters. So the particle mapped in the depth space has a proper row and column index. In this way, we can compare the z of the particle with the z of a point in the depth map at the corresponding row and column of the particle.

Summarizing:

- if occlusions do not occur position and velocity of the object are estimated according to the standard formulation of PF Algorithm 4;
- if the object goes under occlusion the PF runs in open loop Algorithm 5. The particles are projected in the depth space and their z is compared to the z of the corresponding elements in the depth map. If the particle z is less than the one of the pixel z at the same position i, j the particle weight is set to zero.

4.4 Filters on the depth map

Analysing the data provided by the Kinect it is possible to observe that there are different values of depth equal to zero. Zero is the value used by Kinect to indicate invalid data or out of range measurements. Invalid data are mainly related to the lighting condition and imaging geometry. The lighting condition influences the measurement of disparities. In strong light, the laser speckles appear in low contrast in the infrared image, which can lead to outliers or gap in the resulting point cloud. The imaging geometry includes the distance to the object and the orientation of the object surface relative to the sensor. Also the properties of the object surface impact the measurement of points. Smooth and shiny surfaces that appear overexposed in the infrared image impede the measurement of disparities, and imply a gap in the resulting point cloud: see for instance the surface of the table in Figure 4.6.

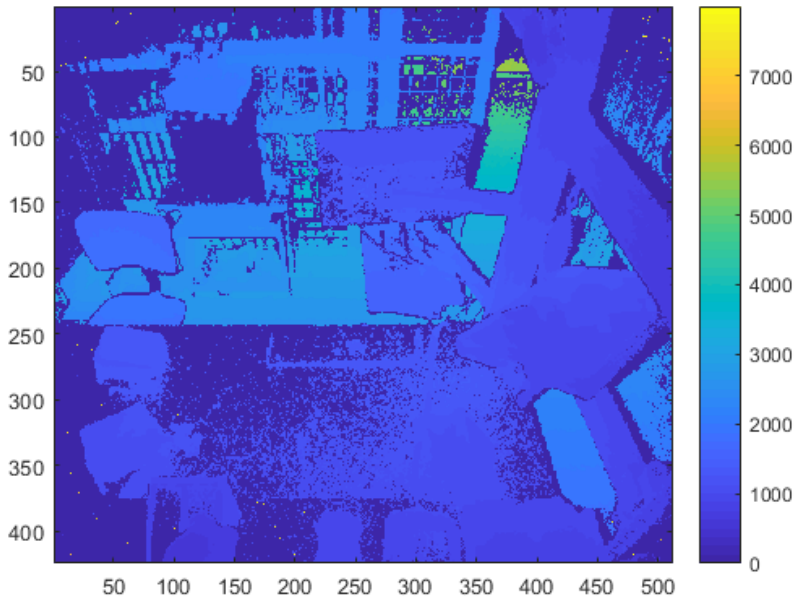


Figure 4.6: *Depth map. At each pixel a colour associated representing its depth value in millimeters: from dark blue which corresponds to zero to yellow representing the max depth measured*

Zero values must be replaced because they lead to an incorrect representation of the surrounding environment and in particular of the possible occlusions. Standard filters such as Gaussian filters or median filters are not suitable for our case. The question now is how to replace these zero pixels in the depth map? To answer this question we experiment two different approaches.

4.4.1 Stretching filter

This filter is intended to replace the zero with the first non-zero value on the left along the row. If this does not exist, the first non-zero value on the right is taken. The name comes from the fact that non-null values are stretched along the row to replace zero values. The algorithm is reported below (see Algorithm 6).

Algorithm 6 Stretching filter

```
1:  procedure StretchingFilter(depthMap)
2:      for i = 1 to n_rows do
3:          if depthMap(i, 1) = 0 then
4:              depthMap(i, 1) = depthMap(find(depthMap(i, :) ≠ 0)
5:              box = depthMap(i, 1)
6:          else
7:              box = depthMap(i, 1)
8:          end if
9:          for j = 1 to n_cols do
10:             if depthMap(i, j) = 0 then
11:                 depthMap(i, j) = box
12:             else
13:                 box = depthMap(i, j)
14:             end if
15:         end for
16:     end for
17:     return depthMap
18: end procedure
```

Consider the example in Figure 4.7 to better understand the stretching filter technique.

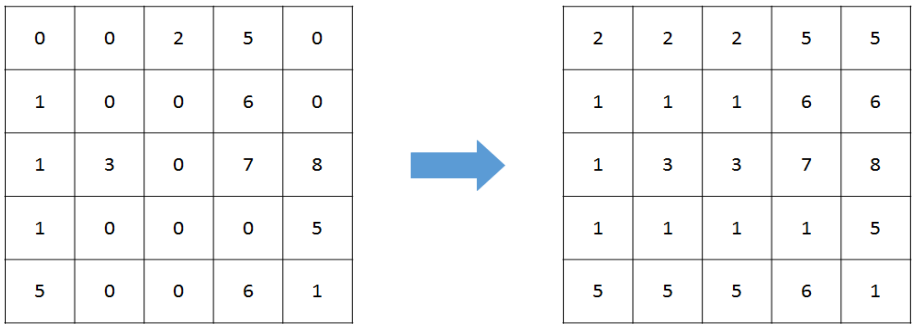


Figure 4.7: *Stretching filter example*

The results after applying the stretching filter on the depth map is shown in Figure 4.8. Note how all the zero values on the table surface and on the image boundaries have been substituted (the chromatic scale do not start from zero). Also the contour of the screen has been better reconstructed.

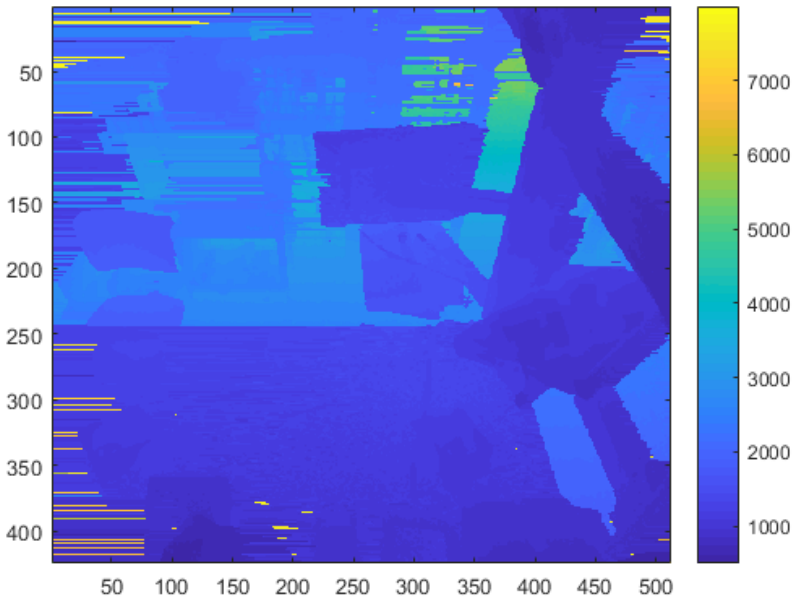


Figure 4.8: *Stretching filter applied on the depth map in* Figure 4.6

4.4.2 Neighbour filter

Instead of substituting zero values with the nearest non-zero value in the row, the neighbour filter substitutes the zeros with the nearest non-zero values in the depth map. To do so the algorithm searches in squares of increasing size until a non-zero value is found. The number of pixels to be checked increases linearly with the level considered, in particular, it grows as $8l$ where l is the level. For instance, in the first level, we have to check at most $3^2 - 1^2$ pixels that is to say $(2l+1)^2 - (2(l-1)+1)^2$ pixels. Therefore if the region of zeros is very large the algorithm is very demanding from the computational point of view. This situation is common at the boundaries of the depth image. The algorithm of the neighbour filter is reported in Algorithm 7.

Once again an example of the working principle of the filter is reported below in Figure 4.9.

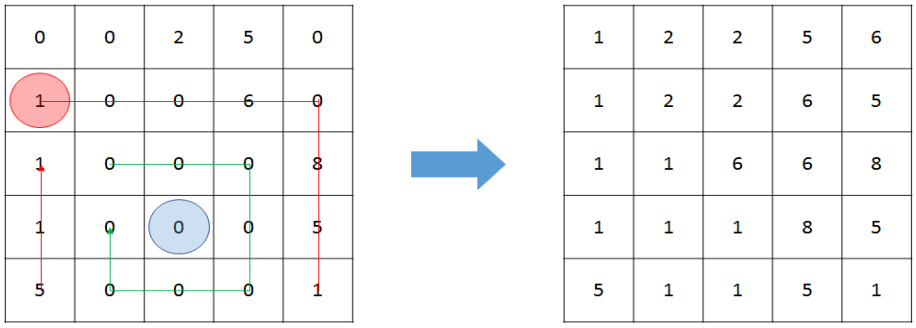


Figure 4.9: *Neighbour filter example: green line indicates the first level, red one indicates the second level*

Algorithm 7 Neighbou filter

```
1: procedure NEIGHBOUR FILTER(depthMap)
2:     for i = 1 to n_rows do
3:         for j = 1 to n_cols do
4:             flag = 0
5:             if depthMap(i, j) = 0 then
6:                 l = 1
7:                 while flag = 0 do
8:                     if ∃e ∈ depthMap at level l: e ≠ 0 then
9:                         dm(i, j) = e
10:                        flag = 1
11:                    end if
12:                    l++
13:                end while
14:            end if
15:        end for
16:    end for
17:    return dm
18: end procedure
```

The result after applying the neighbour filter on the depth map can be seen in Figure 4.10.

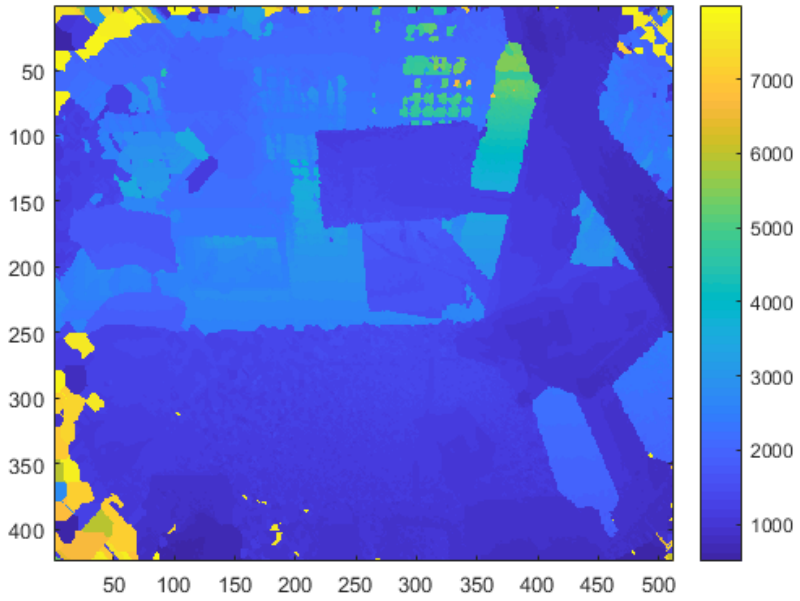


Figure 4.10: *Neighbour filter applied on the depth map in* Figure 4.6

4.4.3 Comparison

Both filters share the strategy of filling the missing information with the near one available. If the stretching filter does it by rows, the neighbour filter search in all directions. In general neighbour filter performs better in the sense that the resulting image is closer to reality. This can be seen for instance looking at the profile of the screen or in how the chair structure is reconstructed (see Figure 4.11).

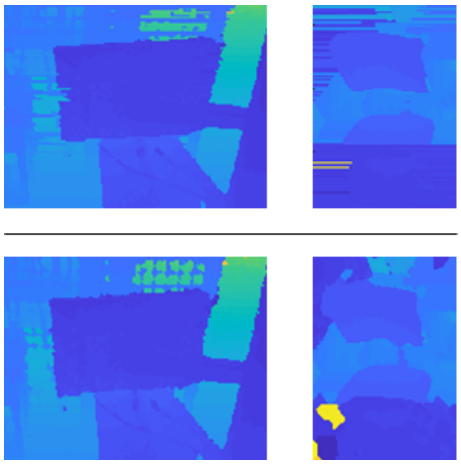


Figure 4.11: *Comparison between the stretching filter (top) and neighbour filter (bottom)*

On the other hand in terms of computational effort, the stretching filter is more than 50 times faster than the neighbour filter. The computation time estimate using MATLAB is approximately of 0.05 s for the stretching filter and 2.5 s for the neighbour filter. To reduce the computational time the filters have been applied only to the pixels of the depth map of interest, that is, those where the particles are projected. This expedient dramatically reduces the computational time needed. Both filters have been tested and a deep analysis of the results will be later discussed in Chapter 5.

4.5 Skeletal tracking

Skeletal tracking is a feature implemented in the Kinect which allows to recognize people and follow their actions. This feature provides the skeletal points also named human joints. Kinect V2 detects up to 25 different skeletal points as we can see in Figure 4.12. In our work we consider only the upper body and in particular we used only the skeletal points relative to the head, spine mid, left shoulder, right shoulder, left elbow, right elbow and left hand and right hand.

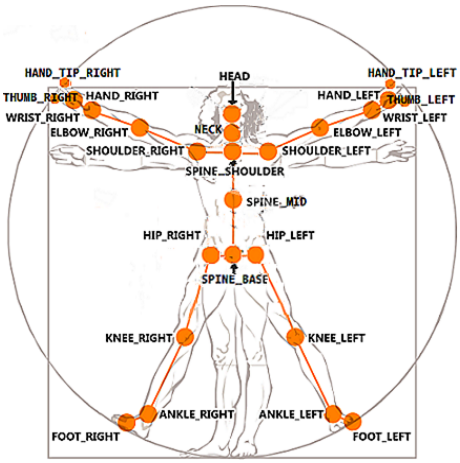


Figure 4.12: *Skeletal points recognized by Kinect*

These points are given in the camera space of the Kinect. Determining the skeletal points is a two stage process:

1. compute the depth map using the structured light;

2. infer body position using machine learning.

This second stage requires the use of a randomized decision forest learned from over a million of training examples.

With Kinect V2 it is possible to track up to six people. Skeleton tracking is designed to recognize users facing the sensor. At each skeletal joint returned a tracking state is associated. The tracking state can take three different values: 0, 1 or 2: 0 stands for joint not tracked, 1 for inferred and 2 for tracked. Value 1 is used by Kinect to denote a partially occluded, clipped or low confidence joint. For our purpose we consider values other than 2 as not valid.

4.6 Cartesian vs joint space

In this section, we will enter the application details regarding the human pose estimation in presence of occlusion.

As already seen in Chapter 2 to describe the human pose we can use two different approaches: the first one uses a kinematic model to represent human motion while the second one uses the skeletal points. Now we will see the pros and cons of pose estimation using these two different approaches.

Consider the pose estimation by joint variables. To define the human pose in joint space we need 12 dofs $p = (x, y, \theta, \rho, \alpha^{right}, \alpha^{left})^t$. Since the Kinect provides the skeletal data in the camera space, to retrieve the measurements we need to perform a kinematic inversion. This procedure has to be done at each frame. Moreover, the kinematic inversion has to be changed according to which skeletal point goes under occlusion. In the joint space, it is very easy to impose constraints on the evolution of particles in order to satisfy physical limits on joint variables like those in [4]. Moreover, we do not need to consider the distance between skeletal points because they can be fixed during the initialization phase.

$$
\begin{aligned}
-9^\circ \leq & \alpha_1 \leq 160^\circ \\
-43^\circ + \frac{\alpha_1}{3} \leq & \alpha_2 \leq 153^\circ - \frac{\alpha_1}{6} \\
-90^\circ + \frac{7\alpha_1}{9} - \frac{\alpha_2}{9} + \frac{2\alpha_1\alpha_2}{810} \leq & \alpha_3 \leq 160^\circ + \frac{4\alpha_1}{9} - \frac{5\alpha_2}{9} + \frac{5\alpha_1\alpha_2}{810} \\
20^\circ \leq & \alpha_4 \leq 180^\circ \\
-30^\circ \leq & \rho \leq 90^\circ
\end{aligned}
$$

However, describing an occlusion in the joint space requires the use of the configurational space (C-space) totally impractical if a space made of 12 dimensions is considered. Let's for instance consider the simple case where the right hand is occluded. Performing the kinematic inversion we can determine all the joint variables except for α_3^{right} and α_4^{right}. So the particles associated with these variables start propagating in open loop. Letting the particles propagate within their joint limits is not sufficient. The admissible joint space is further redefined

by the occlusion itself (Figure 4.13).

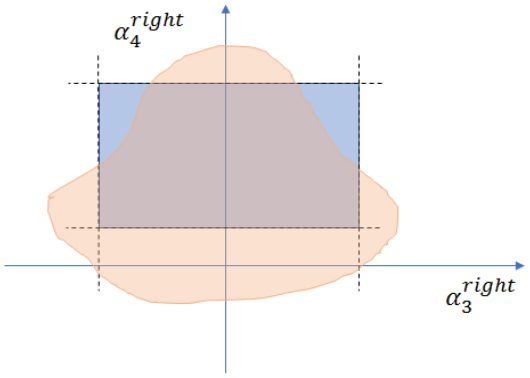


Figure 4.13: *In blue: joint space where* α_3^{right} *and* α_4^{right} *can propagate according to their physical limits. In red: joint space allowed by the occlusion*

Since configurational space is an impracticable way to exclude the particles which violate the occlusion constraints as defined in Section 4.3, the particles must be projected into Cartesian space. This requires a direct kinematic procedure that has to be applied to each particle. It is clear that the number of computations grows linearly with the number of particles.

The alternative is to estimate the human pose through skeletal points. To do so we can track 8 skeletal points which correspond to 24 coordinates, (x, y and z per each skeletal point). No kinematic inversion is needed to run the PF because we can operate directly in the Cartesian space. Imposing constraints on the feasible pose is very hard in the Cartesian space because constraints are naturally expressed as limits on the joint variables. Another disadvantage is related to the skeletal distances that have to be imposed as an additional constraint on the propagation of particles (it will be better explained in the next section). On the other hand, the constraints coming from the occlusions are very trivial to be imposed because also the depth map is expressed in the Cartesian space.

To choose which of the two approaches to use, the following aspects must be considered:

- the algorithm must operate in real time, that is to say with a frame rate not below 20 frames per second;
- even in presence of occlusions the estimate must be consistent with the possible configuration of the human body.

To meet the first requirement it is clear that working in the joint space is a hard approach. In fact to exclude particles inconsistent with the occlusions a direct kinematic is needed. Although the computation time required by this task is rather small, approximately 1.5 ms in MATLAB, considering a huge number of particles it can become the bottle neck to ensure real-time estimation.

Concerning the second aspect, pose estimation in joint space seems to be the best approach thanks to the possibility of imposing limitations on the human pose, difficult to be obtained in the Cartesian space. However looking at Figure 4.14 we realize how unfeasible poses involve a part of the workspace behind the human. Now, considering the scenario where the collaborative robot is placed in front of the human co-worker and the vision sensor on top of that, limiting the possible pose behind the human has little interest. More important is instead to keep the uncertainty limited coherently with the skeletal distances.

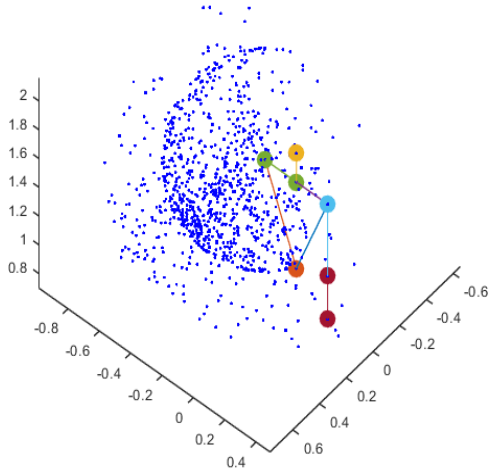


Figure 4.14: *Reachable workspace of the right human arm*

For the above-mentioned reasons, we decided to estimate the pose using the skeletal points. We tracked each skeletal point independently one from the other. The reason why we do so, instead of tracking at the same time all the 8 skeletal points, is that PF suffers from dimensionality problem. When the state to be tracked becomes too big the performance of the PF gets worse in terms of estimation error. This is a well-known problem in the literature, in fact, hierarchical PF approaches have been proposed to face this issue [15]. In this way, we reduced the problem of pose estimation to a problem of tracking multiple point objects in the Cartesian space.

Each point to be tracked is described by the following dynamic equation:

$$s_k = As_{k-1} + \eta_k \tag{4.8}$$

where $s_k = (x_k, y_k, z_k, \dot{x}_k, \dot{y}_k, \dot{z}_k)^t$. The first three components of the state are the position of the skeletal point in space, while the second three are the velocity components. η_k has been modeled as a Gaussian noise $\sim N(0, Q)$. The procedure to estimate Q will be discussed in Section 4.8. Each skeletal point has been

estimated using $N = 500$ particles for a total number of $N_{tot} = 4000$ to estimate the entire pose. The number of particles has been chosen so as to obtain a good compromise between the estimation error and the computation time.

Regarding the output equation, once again we consider:

$$y_k = C_k s_k + \zeta_k \tag{4.9}$$

where $C = (I_{3\times3} \quad O)$ and $\zeta_k \sim N(0, R)$. We modeled it as $R = \sigma_y^2 I$ with a value of $\sigma_y = 0.01\,\text{m}$ greater than the real one. This has been done to avoid degeneracy phenomenon [11], [12].

The skeletal points have been tracked in a reference frame different from the camera frame. We have expressed the estimated pose in the reference frame where the movements of the robot are defined (see Figure 4.15). In order to do this, we make use of a rotation matrix and a translation vector. The relation between the two reference frames is given in equation (4.10):

$$P^{robot} = Rot \cdot P^{camera} + Trn \tag{4.10}$$

$$Rot = \begin{pmatrix} 0 & 0.669131 & 0.743145 \\ 1 & 0 & -1.66533 \cdot 10^{-16} \\ -1.66533 \cdot 10^{-16} & 0.743145 & -0.669131 \end{pmatrix} \quad Trn = \begin{pmatrix} -0.3 \\ 0 \\ 0.755 \end{pmatrix}$$

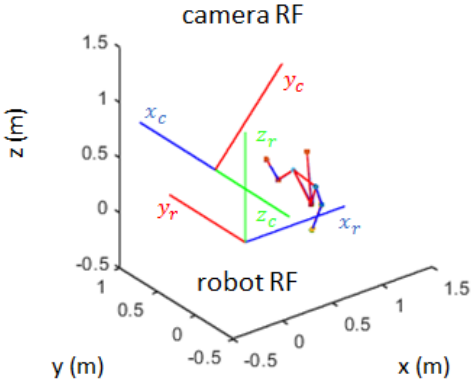


Figure 4.15: *Camera and robot reference systems*

4.7 Implementation of distance constraints

The distances between skeletal points can be imposed as additional constraints on the particles propagation. Before imposing them two steps are required. The first one consists in estimating the distances between the skeletal points. This can be done considering an initialization phase. In this phase, the data incoming from the Kinect relative to the skeletal points can be used to retrieve the skeletal distances for instance by computing the Euclidean norm between points. The second step consists in determining a hierarchical structure in the human skeleton. In order to define such a structure, we limit the possible occlusions to those of the arms (nevertheless the analysis of all the 2^8 possible cases can be done). However, this simplifying hypothesis is reasonable if we consider that in the assembly process the body parts which have a high probability to be occluded are the arms.

Under the previous assumption, the possible situation of occlusions are the following:

1. occlusion of the hand;
2. occlusion of the elbow;
3. occlusion of both the hand and the elbow.

We numbered the skeletal points as in Figure 4.16. The derivation of the hierarchical structure will be made according to this numbering.

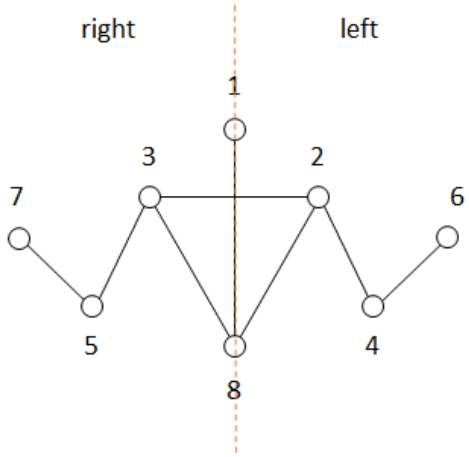


Figure 4.16: *Numbering of the skeltal points*

For the sake of simplicity the analysis of the possible occlusions will be performed only for the skeletal points associated to the left arm. Similar arguments can be used for the right arm.

When only the hand is in occlusion we can use the available information of the elbow to limit the possible position in space of the hand. In fact, the hand can be in the points of space which lie at a distance d_{EH} from the estimated position of the elbow, where d_{EH} is the Euclidean distance between the elbow and the hand computed during the initialization phase (Figure 4.17). Now, since the probability that particles survive along the surface of a sphere is very low, the region of propagation has been modified to assume the shape of a spherical crown. In a similar situation, we define skeletal point number 4 as "father" of skeletal point 6 and we denote this like $4 \succ 6$.

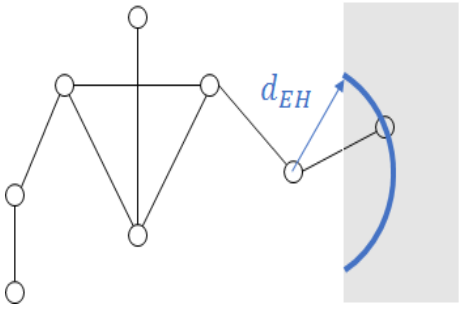


Figure 4.17: *Occlusion of the hand*

When instead only the elbow goes in occlusion, we can say that $2 \succ 4$ but also $6 \succ 4$. In such a way the particles associated to the elbow can propagate in the intersection of the two spherical crowns which have a mean radius d_{SE} and d_{EH} respectively with the obvious meaning of notation (see Figure 4.18).

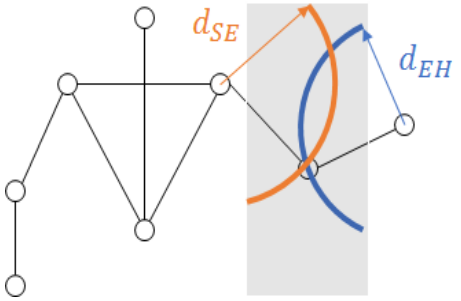


Figure 4.18: *Occlusion of the elbow*

Finally, the case where both hand and elbow are in occlusion is tackled. In this case, the particles associated to the elbow can propagate in a spherical crown centered in the estimated position of the shoulder with mean radius d_{SE}, while the ones related to the hand can propagate inside the sphere of radius $d_{SH} = d_{SE} + d_{EH}$ once again centered in the shoulder. We design such a constraint on the hand position to take into account position close to the shoulder one. We describe this situation according our formalism in this way: $2 \succ 4$ and $2 \succ 6$ (refer to Figure 4.19).

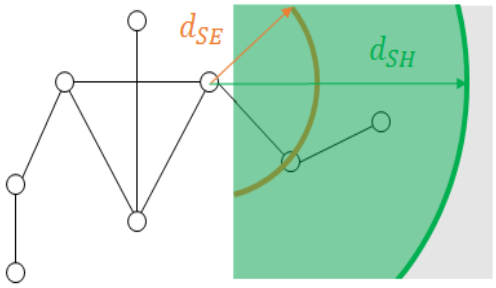


Figure 4.19: *Occlusion of the hand and the elbow: in green the area where particles of the hand can propagate; in red the spherical crown where particles of the elbow can propagate*

4.8 Process and measurement noise estimation

The performance of the PF depends heavily on the covariance matrices R and Q associated with the measurement and the process noise, respectively. In particular, the covariance matrix of the process noise is crucial to define the behaviour of the particles in open loop. In the presence of occlusions, our goal would be to propagate the particles in a way consistent with the dynamics of human movements. To achieve this goal we have to estimate the process noise. Generally speaking, the covariance matrix of process noise is harder to be determined than that of the measurement noise by routine experiments, since the statistical property of process noise cannot be obtained directly by collecting a large number of sensor data due to the intrinsic coupling of process noise and system dynamics. To estimate process noise we use a recursive covariance estimation (RCE) algorithm. This algorithm has been taken from [21] and reformulated to fit our case study.

First of all, we estimate the measurement noise covariance matrix. To do this we collect the position measurements of a stationary skeletal point in space. Considering a data set made of 300 samples we retrieve the variance in the three spatial directions. The results are shown in Table 4.1.

$\sigma_X^2 \quad (m^2)$	$\sigma_Y^2 \quad (m^2)$	$\sigma_Z^2 \quad (m^2)$
$4.26 \cdot 10^{-5}$	$2.89 \cdot 10^{-5}$	$2.43 \cdot 10^{-5}$

Table 4.1: *Variance on the x, y, z direction*

Making the assumption that the noise is equal along all directions we average the variances and we obtain a variance of $3.19 \times 10^{-5}\,\mathrm{m}^2$.

Under the previous assumption, to estimate the process noise covariance, we can consider the following simplified system:

$$s_{k+1} = As_k + \eta_k$$
$$y_k = Cs_k + \zeta_k$$

where the state is $s_k = (x_k, \dot{x}_k)^t$ and the matrices A, B and C are formed in this way:

$$A = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \qquad B = I_{2\times 2} \qquad C = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

Our goal is to retrieve the terms of matrix Q which in our case is:

$$Q = \begin{pmatrix} q_1 & q_2 \\ q_3 & q_4 \end{pmatrix}$$

Introducing the backward shift operator z^{-1}, it is possible to rewrite y_k as function of the process and measurement noise:

$$y_k = C(I - Az^{-1})^{-1}B\eta_{k-1} + \zeta_k \tag{4.11}$$

Performing the left coprime factorization we can express (4.11) in the form:

$$y_k = C(I - Az^{-1})^{-1}B\eta_{k-1} + \zeta_k = a^{-1}(z^{-1})b(z^{-1})\eta_{k-1} + \zeta_k \tag{4.12}$$

where $a(z^{-1})$ and $b(z^{-1})$ are polynomial matrices with the following structure:

$$a(z^{-1}) = a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}$$
$$b(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}$$

In our case $n = 2$ (it represents the dimension of the state) and the polynomial matrices take this form:

$$a(z^{-1}) = 1 - 2z^{-1} + z^{-2} \tag{4.13}$$
$$b(z^{-1}) = (1 \quad 0) + (-1 \quad \Delta t)z^{-1} + (0 \quad 0)z^{-2} \tag{4.14}$$

Multiplying both sides of equation (4.12) by $a(z^{-1})$ we get:

$$a(z^{-1})y_k = b(z^{-1})\eta_{k-1} + a(z^{-1})\zeta_k \tag{4.15}$$

Let's call $\xi_k = a(z^{-1})y_k$, $W_k = b(z^{-1})\eta_{k-1}$ and $V_k = a(z^{-1})\zeta_k$. Under the assumption that process and measurement noise are uncorrelated and $E(\xi) = 0$, $E(W) = 0$, $E(V) = 0$ it holds that $Cov(\xi) = Cov(W) + Cov(V)$:

$$E[\xi_k \xi_{k-j}^t] = \sum_{i=j}^{n} a_i R a_{i-j}^t + \sum_{i=j}^{n} b_i Q b_{i-j}^t \quad \forall j = 0, ..., n \tag{4.16}$$

Since R is know, we computed it previously (Table 4.1), and the $Cov(\xi)$ can be computed by the measurements, we can retrieve the covariance matrix of the process noise Q. Hence we can transform the equation (4.16) into an equivalent linear system $\Omega Vec(Q) = \theta$. Recalling that $Vec(\cdot)$ is the vectorization operator the solution of this system is:

$$Vec(Q) = \Omega^{\dagger}\theta \tag{4.17}$$

where $\Omega^{\dagger}$ denotes the Penrose-Moore pseudoinverse of Ω. The complete derivation of the matrix Q is reported in Appendix B.

We performed an experiment collecting a set of 1000 data samples of the right hand motion, acquired using a sampling time of $\Delta t = 0.04\,\mathrm{s}$ and we obtained the results in Table 4.2 for q_1, q_2 ($q_3 = q_2$) and q_4.

$q_1 \quad (m^2)$	$q_2 \quad (m^2/s^2)$	$q_4 \quad (m^2/s^2)$
$7.18 \cdot 10^{-4}$	$-4.10 \cdot 10^{-10}$	$6.26 \cdot 10^{-9}$

Table 4.2: *Terms of the process noise covariance matrix Q*

Since the values of the off-diagonal terms are one order of magnitude less than the smallest term on the diagonal ($q4$), we neglect them considering a diagonal Q matrix.

4.9 General structure of the algorithm

To provide an overview of the algorithm designed, the fundamentals steps are reported in Algorithm 8.
After an initialization phase where the matrices A, C, Q and R are defined the algorithm enters a state machine. As long as there are no humans in the scene the algorithm remains in state 1 and the scene is continuously updated.

When a human enters the scene we move to the second state. Here the skeletal distances are computed and the initial conditions of the PF are imposed.

After the initialization phase, the system enters the state number 3. Here at each iteration the depth map and the humans in the scene are updated. Then the skeletal points and their tracking state are acquired. According to which skeletal point is under occlusion, the hierarchy of the skeletal points is defined as we have previously described in Section 4.7. For each skeletal point, the PF is run. If the skeletal point is not in occlusion the closed loop formulation of the PF is executed, otherwise the open loop formulation is performed. In the latter case, the particles not consistent with the depth map and the skeletal distances are eliminated.

Algorithm 8 Pose estimation under occlusion

```
1: procedure POSE ESTIMATION UNDER OCCLUSIONS
2:     PF_ParametersInitialization(A, C, Q, R)
3:     ∀ frame, begin state machine:
4:     if state = 1 then
5:         UpdateHumans()
6:         if ∃ human in the scene then
7:             state = 2
8:         end if
9:     else if state = 2 then
10:        ComputeSkeletalDistances()
11:        AcquireSkeletalPoints()
12:        s0 = InitializeStateEstimate()
13:        s_P = InitializeStateParticles()
14:        state = 3
15:    else if state = 3 then
16:        UpdateDepthMap()
17:        UpdateHumans()
18:        AcquireSkeletalpoints()
19:        AcquireTrackingState()
20:        DefineTheHierarchyAccordingToTheOcclusion(tracking_states)
21:        for i =1 to n_skeletal_points do
22:            PF(data_skeletal_points, father):
23:            if tracking_state = 2 then
24:                Closed loop formulation of PF
25:            else
26:                Exclude the particles not consistent with the depth map
27:                Exclude the particles which violate the distance constraints
28:            end if
29:            endPF
30:        end for
31:    end if
32: end procedure
```

Chapter 5

Simulation and experimental results

5.1 Introduction

In this chapter, we present the results achieved in this thesis. First of all in Section 5.2 we show the results obtained using the MATLAB simulation environment. Here the tracking capabilities of PF algorithm are discussed and a simulation of the behaviour under occlusion is presented. The chapter continues with the experimental results obtained by processing the data provided directly by the Kinect. Finally, a demonstration of the algorithm performance when applied to a real human-robot collaborative task is shown.

5.2 Simulations

5.2.1 Particle Filter performance

First of all the tracking capabilities of the PF algorithm have been tested. In this phase, no occlusions have been considered. To do so we simulate a motion of the human body and we compute the estimation error of the skeletal points position. The simulated motion has been defined in the joint space according to

the equations below, describing a walking motion.

$$
\begin{aligned}
&fps = 25 \\
&x(k) = 2k/fps \\
&y(k) = 0 \\
&\theta(k) = 0 \\
&\rho(k) = \frac{\pi}{20} + \frac{\pi}{20} sin(4k/fps) \\
&\alpha_1^{right}(k) = \frac{\pi}{20} sin(k/fps) \\
&\alpha_2^{right}(k) = \frac{\pi}{5} cos(2k/fps) \\
&\alpha_3^{right}(k) = 0 \\
&\alpha_4^{right}(k) = -\frac{\pi}{6} - \frac{\pi}{4} cos(2k/fps) \\
&\alpha_1^{left}(k) = \frac{\pi}{20} cos(k/fps) \\
&\alpha_2^{left}(k) = -\frac{\pi}{20} cos(2k/fps) \\
&\alpha_3^{left}(k) = 0 \\
&\alpha_4^{left}(k) = -\frac{\pi}{6} + \frac{\pi}{4} cos(2k/fps)
\end{aligned}
$$

We obtain the measured values of the skeletal points by adding a Gaussian noise to the true positions. We used a covariance matrix of the noise equal to $R = \sigma_y^2 I$ with $\sigma_y^2 = 3.19 \times 10^{-5}\,\mathrm{m}^2$. In the PF algorithm we used instead a $\sigma_y^2 = 1 \times 10^{-4}\,\mathrm{m}^2$ to avoid degeneracy problems. The estimation error has been computed like the Euclidean distance between the true skeletal position and the estimated one. The results are shown below in Figure 5.1.

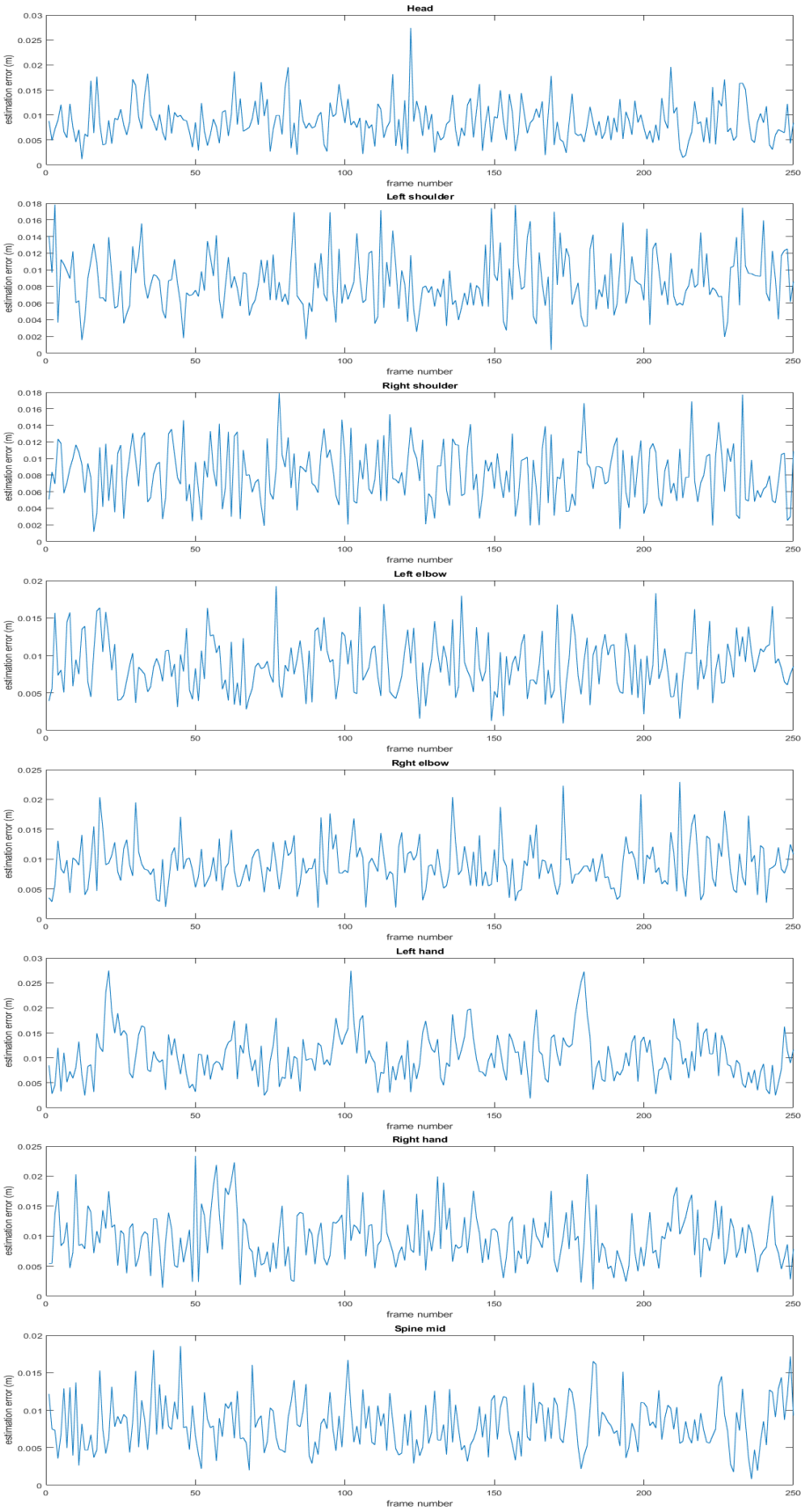


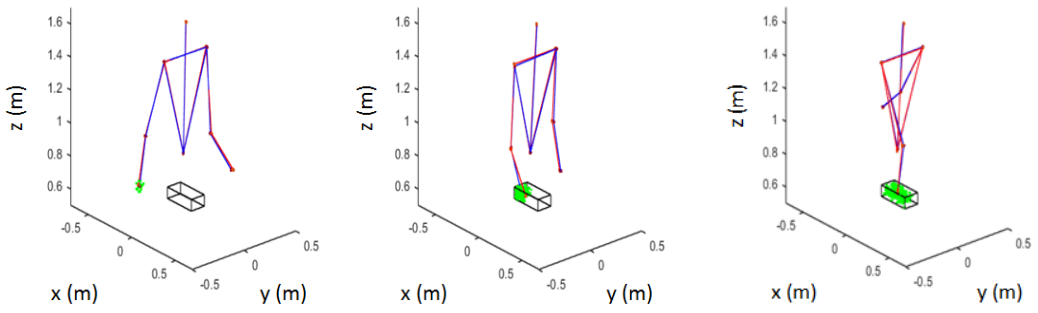
Figure 5.1: *Estimation errors on the skeletal points position*

Analysing the data we retrieve a maximum estimation error equal to 2.75 cm and a mean error approximately of 9 mm.

5.2.2 Simulation of an occlusion

Then we tested the algorithm in an occlusion situation where the shape of the occlusion was a priori known. We simulate an operation where the operator's right hand goes into occlusion with a box-shaped object.

First, the constraints on the distances were not considered. The set of particles associated with the right hand after the resampling phase are represented in green in the figures below. Figure 5.2 shows the situation before (a) and during the occlusion (b), (c) .



(a) (b) (c)

Figure 5.2: *Propagation of particles in an occlusion situation. In green, we can see the particles associated with the right hand, in red the true pose while in blue the estimated one*

The same situation has been tested considering also the distance constraint imposed on the propagation of particles. We can see from Figure 5.3 that the particles propagate only in the intersection of the spherical crown and the region of occlusion. The same situation will be presented next while considering a real experiment.

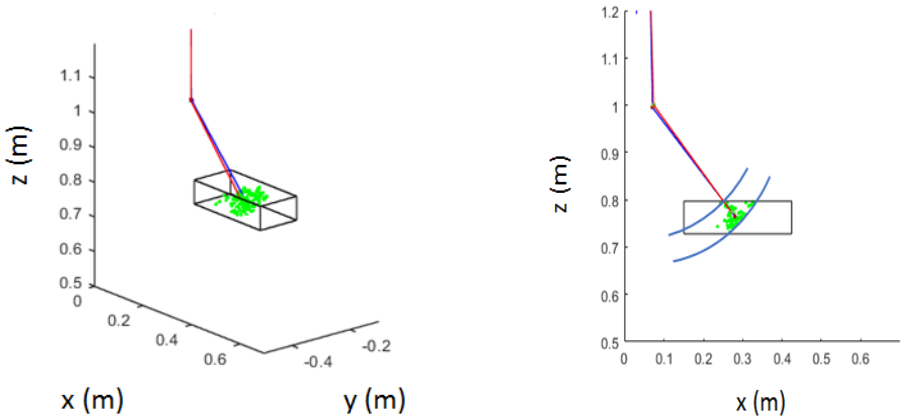


Figure 5.3: *Distance constraint on the right hand particles*

The other possible situation of occlusion considered are reported in the next section.

5.3 Experimental results

Now we present the experimental results obtained using the data from the Kinect sensor. This subsection is structured as follows:

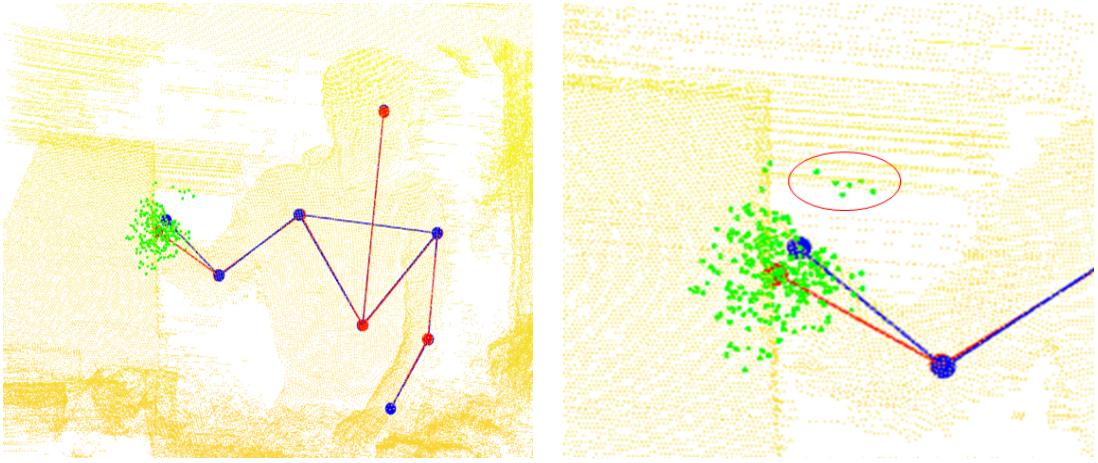
- in the first part we compare the results obtained when an occlusion of the hand occurs applying first the stretching filter and then the neighbour filter (no distance constraints have been taken in account in this phase);
- in the second part we present the cases of occlusion studied relative to the occlusions of shoulder, elbow, and hand;
- finally, the situations of miss and false detections are shown.

Comparison between stretching filter and neighbour filter

Figure 5.4 and Figure 5.5 report a situation where the hand goes in occlusion with a panel. We can see in blue the skeletal points measured by the sensor, in red the

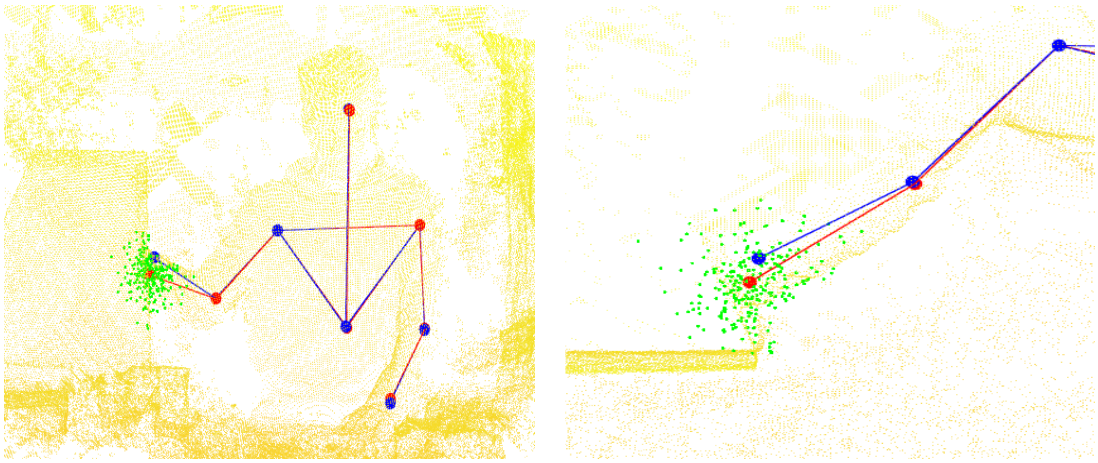
estimated one and in green, the particles associated to the hand are represented. In Figure 5.4 we have the results when the stretching filter algorithm is applied to the depth map while in Figure 5.5 the results when instead the neighbour filter is applied. As we can see from the figures below the neighbour filter performs better in terms of image reconstruction. Note how the panel profile is distorted when the stretching filter is applied. Due to this distortion, some particles survive also out of the occlusion. Those particles have been highlighted by a red ellipse in Figure 5.4 (b). In order to get the whole images, the filters have been applied to the entire depth map. However, remember that the filtering operation of the image is only performed on the points of the map in which the particles are projected to.

Since the neighbour filter performs better, from now on all the results that we will present have been obtained using this filter.



(a) (b)

Figure 5.4: *Example of hand occlusion when the stretching filter is applied on the depth map*

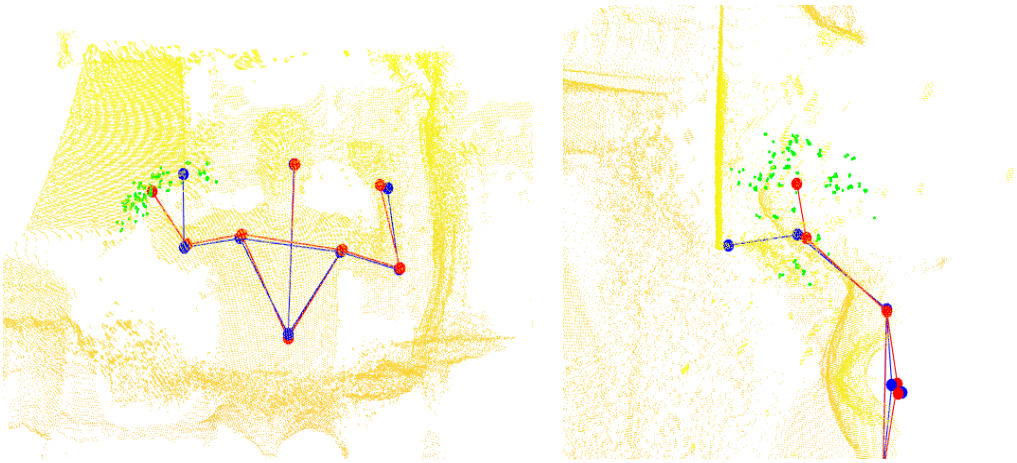


(a) *Frontal view* (b) *Top view*

Figure 5.5: *Example of hand occlusion when the neighbour filter is applied on the depth map*

Cases of occlusion studied

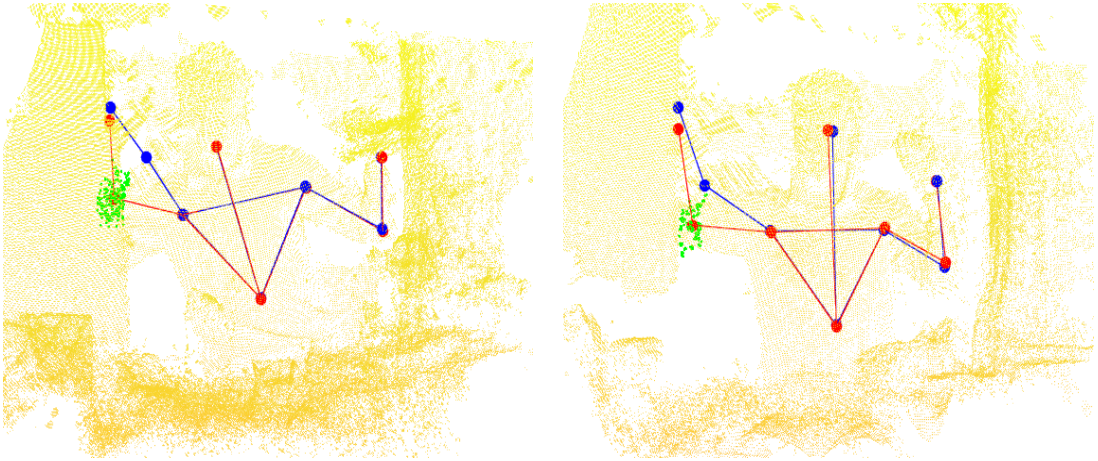
Below are the results regarding the cases of occlusion considered.
First, let's analyse the case when the hand goes into occlusion. Refer to Figure 5.6. Once again in blue, we have the skeletal points measured by the Kinect and in red the estimated ones (from now on we consider this convention of colours). We can see how the measured value of the hand position has been placed at the border of the occlusion and it is completely inconsistent with the forearm configuration. Note how the particles propagate only in the spherical crown centered in the estimated position of the elbow. Moreover, as we can see from Figure 5.6 (b) the particles remain behind the occlusion.



(a) *Front view* (b) *Top view*

Figure 5.6: *Occlusion of the hand*

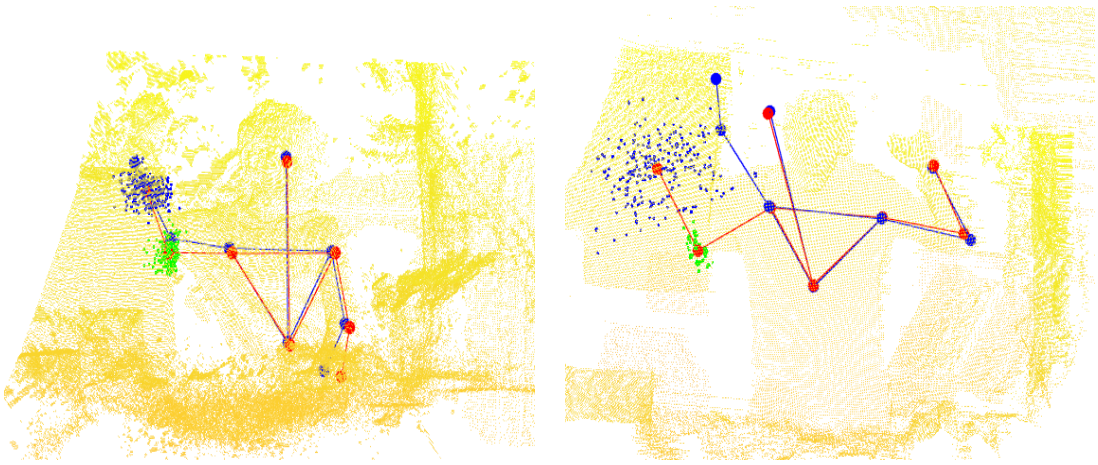
The case when only the elbow is in occlusion is reported below in Figure 5.7. Given the available information on the hand and shoulder position, we can see how the particles associated with the elbow remain limited in a restricted volume in space. In both figures, the estimated pose is closer to the real pose than the measured one. Once again we can observe the tendency of the Kinect to place the measured value at the border of the occlusion.



(a) (b)

Figure 5.7: *Occlusion of the elbow*

Finally, the results when both the hand and the elbow are in occlusion are shown (Figure 5.8). Note how the particles propagate according to the way described in Section 4.7: those of the wrist remain inside the spherical crown centered in the estimated position of the elbow while those associated to the hand remain inside the sphere of radius δ_{SH} centered in the shoulder.



(a) (b)

Figure 5.8: *Occlusion of both the hand and the elbow*

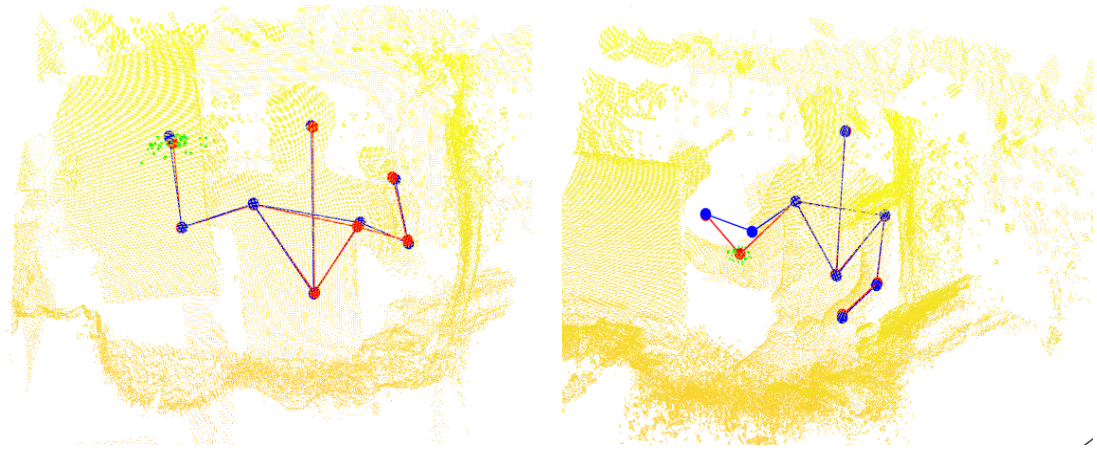
Particular cases

Now we want to show some particular situations. More in depth we will show:

- false detection;
- miss detection.

A pair of examples belonging to the first category is represented in Figure 5.9. In Figure 5.9 (a) the operator has both the hand and the elbow occluded by a panel, while Kinect interprets as occluded only the hand. Another example of false detection is shown in Figure 5.9 (b): the Kinect provides a valid measure of the right hand (tracking state = 2) but, as we can see, it is totally inconsistent with the true hand position. These situations are particularly difficult to be

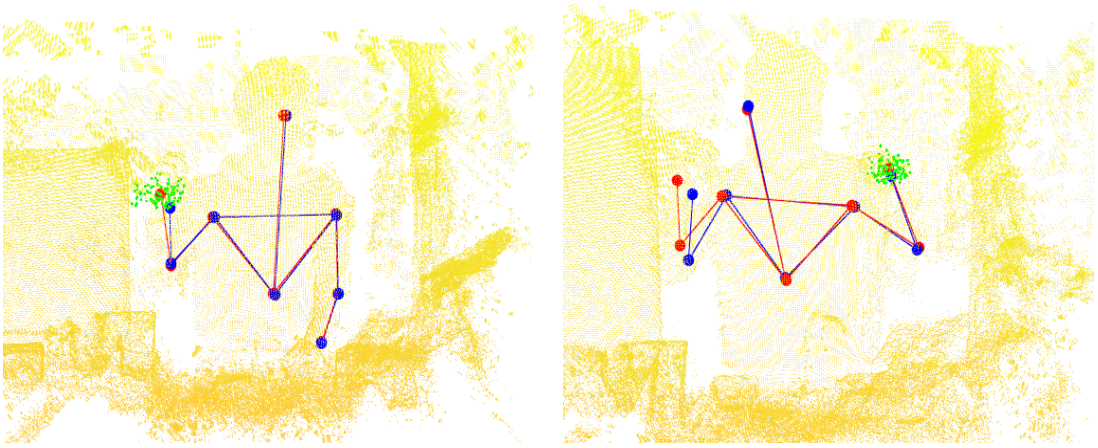
handled since they are related to the identification process of the skeletal points used by the sensor.



(a) (b)

Figure 5.9: *Situation of false detection*

As far as the missed detection is concerned, we can see some examples in Figure 5.10. In these situations the Kinect loses some skeletal points despite they are visible. Miss detection is not so crucial as the false detection, in fact imiss detection is treated as it were an occlusion situation.



(a) (b)

Figure 5.10: *Situations of miss detection*

5.4 Test with YuMi

The pose estimation is the starting point for the computation of the so-called swept volumes i.e. the entire volume that the human can reach given a specific kinematic configuration and a chosen time horizon. Knowing the volume occupied at each time instant and predicting the human's intentions are two fundamental elements of the safe trajectory generation process.

Consider now a scenario in which human and robot cooperate in an assembly process. It is evident that limiting the uncertainty relative to the human pose improves productivity since the robot will follow trajectory closer to the optimal ones. To prove this we performed an experiment in which we compared the estimation technique based on the KF and our proposed technique. How the experiment has been structured is explained in the following.

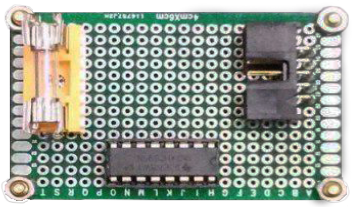


Figure 5.11: *Electronic board*

First, we defined an assembly operation. This operation consists in assembling some electronic components over a printed circuit board (Figure 5.11). Consider the workstation in Figure 5.12: the workstation is represented from the point of view of the human co-worker. In the picture, we can see the robot YuMi (Y) and the Kinect sensor (K) on top of that. In addition, there are two slides: one which feeds the robot (1) and a second one that receives the boards worked by the robot (2). There are also three boxes: in the red one, there are the pieces that have to be assembled (4), in the gray one the worked boards (6) are collected while in the green box there are the worked boards ready for the next assembly

phase. In the process of assembling occlusions occur. The first one is generated in correspondence with the object in (5) while the second one by the panel in (7).

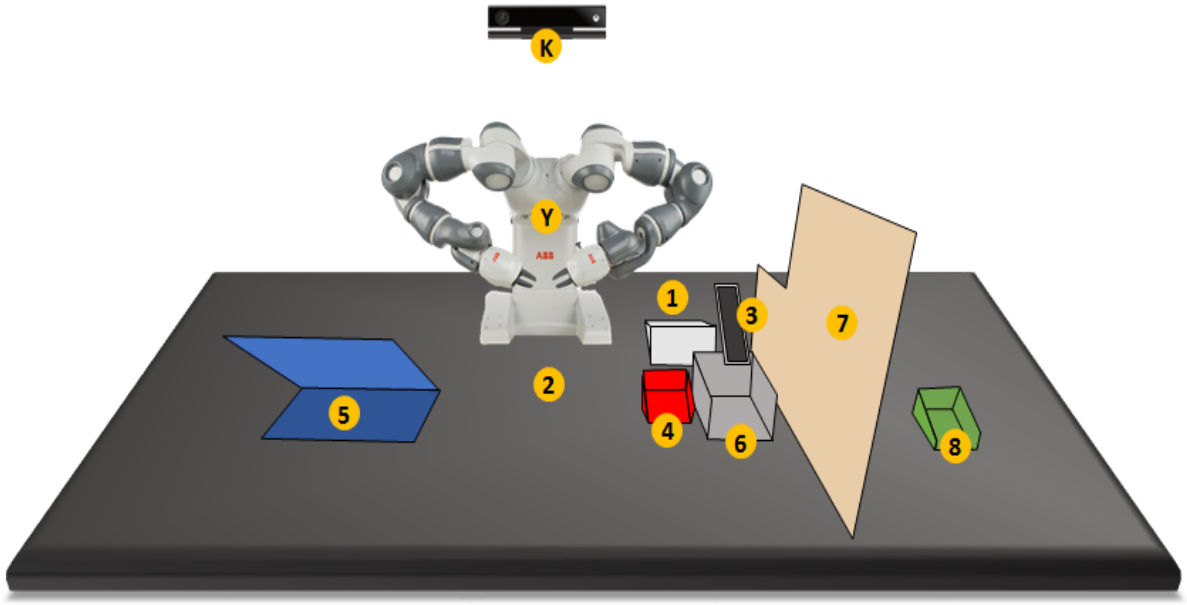


Figure 5.12: *Workstation*

The operations performed by the robot and the human are the following ones: Robot operations:

1. the left arm picks a board from the slider (1);
2. it carries the board in the position (2);
3. when the left arm leaves the board the right arm goes in (2) and assembles a component on the board;
4. the left arm goes back in (2) and picks up the board ;
5. finally the board is brought to the slide (3).

Human operations:

1. picks a board from the box (4);
2. takes a fuse from (5) and assembles it on the board;

3. places the board on the slide (1);
4. takes the worked board from the gray box (6) and leaves it in the green one (8).

To prove how uncertainty about the pose influences the productivity we consider the following experiment. For both the estimation techniques a swept volume on the estimated pose and its uncertainty was built. In particular, we consider a confidence level of 90%. We defined a virtual plane in front of the robot. If the swept volume associated with an occluded skeletal point goes beyond the virtual plane, the speed of the robot is reduced to the 5% of its nominal speed. Some remarks:

- the KF was built to estimate the position of the skeletal points not the value of joint variables. This is a simplifying assumption since in real application KF estimates the joint variables. However, this assumption does not preclude the validity of the conducted experiment since an uncertainty on the joint variables is reflected in an uncertainty on the skeletal position;
- to determine the swept volume including the uncertainty we considered for KF the ellipses associated to each skeletal point built according to $e_x^t \Sigma^{-1} e_x = \chi_\alpha^2$ where e_x is the state error vector and α is the confidence level considered; for the PF we reduce the speed of the robot when the 10% of the particles associated at any skeletal point crosses the virtual plane.;
- when the swept volume at the current time instant crosses the virtual plane the limitation affects only the speed not the path of the trajectories followed by the robot.

We record the time taken by the robot to perform a cycle. One cycle is defined as the sequence of operations from a pickup board from slide (1) action to the next one. We collect the time data relative to 30 robot cycles per each technique considered. The results are shown in Figure 5.13.

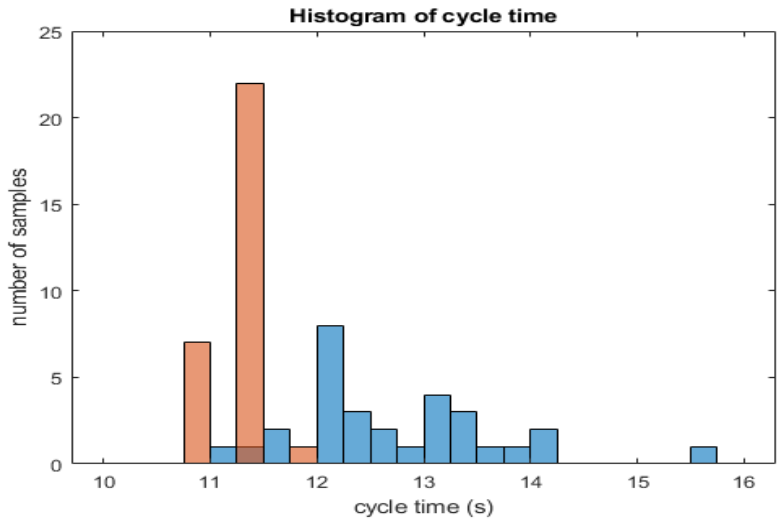


Figure 5.13: *Histogram relative to the robot cycle time*

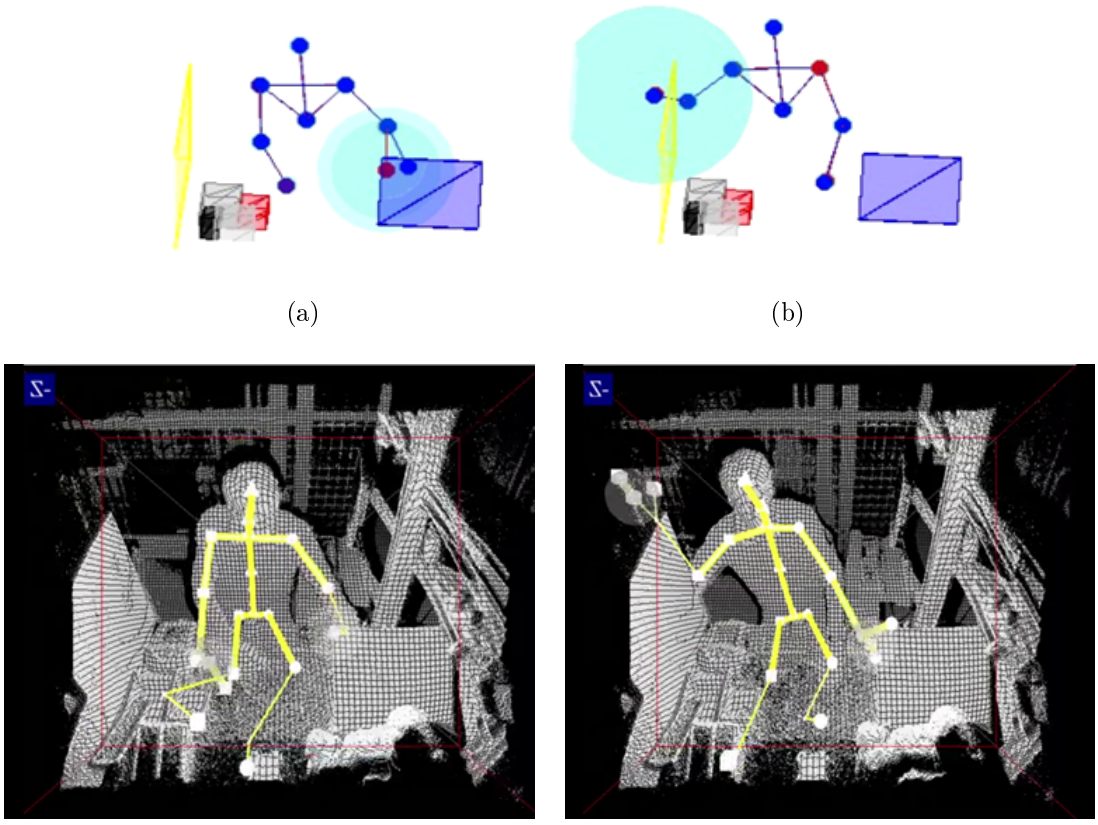
Table 5.1 reports the mean cycle time (MCT).

MCT no filter (s)	MCT KF (s)	MCT PF (s)
11.18	12.74	11.27

Table 5.1: *Mean cycle time*

The MCT when no filters are applied i.e. when the robot always proceeds at nominal speed, is very close to the MCT obtained when the PF is applied. We can observe a decrease in the MCT of approximately 13% when our developed technique is applied. This reduction in the cycle time clearly depends on how long the operator is in occlusion.

Pictures regarding the experiment with the KF are reported in Figure 5.14 (a) and (b): we can see in blue the measured pose and in red the estimated one. At each skeletal point, an ellipse is associated representing the confidence region. We can see how the ellipses diverge when occlusions occur. In Figure 5.14 (c) and (d) we have the depth data and the skeletal joints extracted by the Kinect.

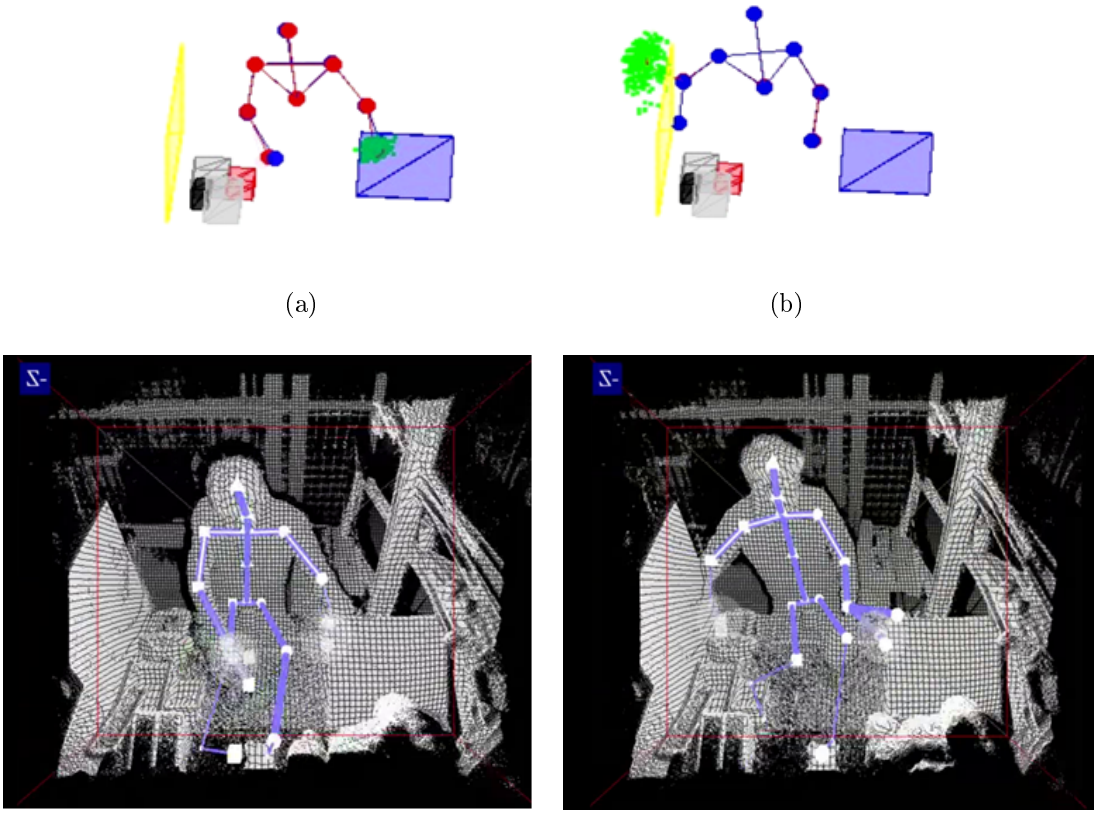


(a) (b)

(c) (d)

Figure 5.14: *Frames from the experiment performed by using the KF*

From the experiment performed using the PF based technique we can see in both Figure 5.15 (a) and (b) how the uncertainty related to the position is confined in space. Moreover particles (in green) remain behind the occlusions.



(a) (b)

(c) (d)

Figure 5.15: *Frames from the experiment performed by using the PF base technique*

The video showing the comparison between KF and our developed technique is available at `https://www.youtube.com/user/MERLINpolimi`.

Summarizing: from the experimental results obtained is evident how the estimation technique developed is able to limit the uncertainty relative to the pose in a way consistent with the shape of the occlusions and the skeletal distances. However, the performance in terms of estimation error highly depends on the skeletal

recognition algorithm implemented in the Kinect. The benefits that our developed technique could bring in terms of productivity have been proven through the demo with YuMi. The benefits have been quantified in an increase of the 13% in productivity, but this improvement depends on the application considered.

Chapter 6

Conclusions

In this work, we presented a technique to estimate the human pose in presence of occlusions. In particular, our aim was to overcome the limits of the currently used estimation technique: the Kalman filter. Our starting point was the analysis of the KF and in particular its behaviour when occlusions occur. From this analysis, the complexity to impose constraints on the state estimate and the impossibility to approximate distribution different from the Gaussian one emerged.

For these reasons, we continued with a research of alternative estimation techniques which best suit our problem (Section 2.6). Among these, we chose the particle filter. This technique is characterized by approximating the state estimate by a set of samples called particles. The sample-based nature of the PF has allowed to introduce, in a very effective way, constraints on the state estimate. These constraints have been exploited to model the surrounding environment (including the occlusions) and to better use the available information on the human pose.

In Chapter 4 we entered the implementation of our technique. More in depth we explained how the depth data from Kinect has been used to model occlusions and in which way we structured our algorithm.

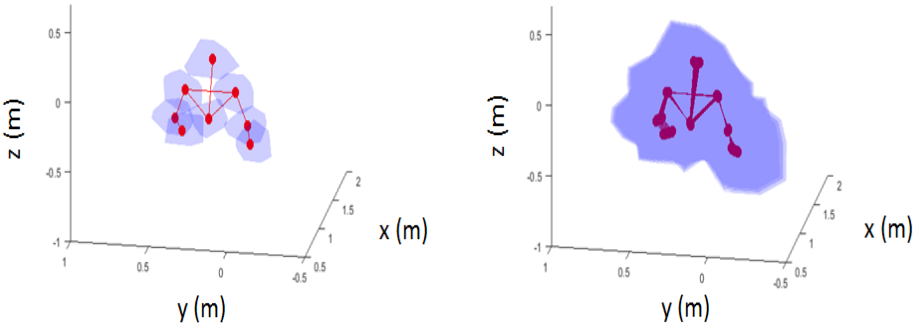
Finally we presented the result achieved. First, we presented the simulation

results obtained in MATLAB. Then we implemented the algorithm to work with Kinect and a series of images from the real application has been shown. We concluded this work performing a demo of a human-robot collaboration to prove how limiting pose uncertainty increases productivity (Section 5.4).

However, we identified some limits in our technique which mainly depend on the data retrieved by Kinect. In particular, the skeletal positions retrieved by the sensor are subjected to the phenomenon of miss detection and false detection. Despite the first one is easily manageable, the second one is more difficult to be tackled. In fact, false detections directly depend on the process of skeletal recognition performed by the Kinect. This process is particularly complex since it is based on training an algorithm to detect the human body in different scenarios and involves techniques of artificial intelligence. This sort of process has been kept out our field of work for reasons of time and complexity.

6.1 Future developments

Pose estimation is one of the numerous aspects necessary to the human-robot interaction. As already said, knowing the position of the human is at the basis of the computation of the swept volumes and thus the generation of safe interaction trajectories. The results achieved by this work can be integrated into the process of the swept volumes computation. If up to now these were computed starting from the estimated value of position and velocity expressed in the joint space, now they must be computed starting from the information in the Cartesian space. An attempt to do so has already been done in the last stages of this work approximating the swept volumes with the volume occupied by the particles propagated in open loop (see Figure 6.1). However, this topic must be definitely deepened.



(a) (b)

Figure 6.1: *Swept volumes evolution in* $5 \cdot \Delta t$

In addition, position uncertainty can be further limited if the information concerning human's intentions and the nature of collaborative action are taken into account in the estimation process. Finally, this work can be easily adapted to work in a sensor redundancy framework.

Appendix A

Bayes estimation

Consider the following discrete time system:

$$x_k = f_k(x_{k-1}, u_k) + v_k \tag{A.1}$$

$$y_k = h_k(x_k) + w_k \tag{A.2}$$

where x_k, u_k, y_k are state, input and output respectively, while v_k and w_k are the process and the measurement noise possibly non-Gaussian; $f_k(\cdot), h_k(\cdot)$ are nonlinear functions. equation (A.1) is the state equation and equation (A.2) is the output equation.

In the following we will consider the model to satisfy Markov assumptions [22], that is to say that past and future data are independent if one knows the current state x_k: in particular we have that $p(x_k|x_{0:k-1}, y_{1:k-1}, u_{1:k}) = p(x_k|x_{k-1}, u_k)$ and $p(y_k|x_{0:k}, y_{1:k-1}, u_{1:k}) = p(y_k|x_k)$.

The goal of Bayesian estimation is to construct the conditional a posteriori distribution $p(x_k|y_{1:k}, u_{1:k})$ given the one at the previous time instant, the input variable u_k and the incoming measurement y_k [23]. The solution is obtained by solving in a recursive way the Bayes' rule:

$$p(x_k|y_{1:k-1}, u_{1:k}) = \int p(x_k|u_k, x_{k-1})p(x_{k-1}|y_{1:k-1}, u_{1:k})d_x \tag{A.3}$$

$$p(x_k|y_{1:k}, u_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1}, u_{1:k})}{p(y_k|y_{1:k-1}, u_{1:k})} \tag{A.4}$$

$$p(y_k|y_{1:k-1}, u_{1:k}) = \int p(y_k|x_k)p(x_k|y_{1:k-1}, u_{1:k})d_x \tag{A.5}$$

Let's denote the conditional posterior distribution as $post(x_k)$, then it is possible to define a recursive algorithm in order to compute the posterior at each time instant (Algorithm 9 taken from [11]):

Algorithm 9 Bayes filter

1: **procedure** BAYES FILTER($post(x_{k-1}), u_k, y_k$)
2: **for** all x_k **do**
3: $\overline{post}(x_k) = \int p(x_k|u_k, x_{k-1})post(x_{k-1})d_x$
4: $post(x_k) = \frac{p(y_k|x_k)\overline{post}(x_k)}{p(y_k|y_{1:k-1}, u_{1:k})} = \eta p(y_k|x_k)\overline{post}(x_k)$
5: **end for**
6: **return** $post(x_k)$
7: **end procedure**

Appendix B

Recursive Covariance Estimation

In the following are reported all the passages needed in order to retrieve the terms of matrix Q (these passages are not reported in [21]).
We can rewrite equation (4.16) with the terms depending on Q at the left side of the equation:

$$\sum_{i=j}^{n} b_i Q b_{i-j}^t = E[\xi_k \xi_{k-j}^t] - \sum_{i=j}^{n} a_i R a_{i-j}^t \quad \forall j = 0, ..., n \tag{B.1}$$

We can rewrite the first member of equation (B.1) as

$$\sum_{i=j}^{n} b_i Q b_{i-j}^t = b_j Q b_0^t + b_{j+1} Q b_1^t + \cdots + b_n Q b_{n-j}^t \tag{B.2}$$

Consider for instance the first term of this sum. We can rewrite it as follows:

$$b_j Q b_0^t = \begin{pmatrix} b_j(1) & b_j(2) \end{pmatrix} \begin{pmatrix} q_1 & q_2 \\ q_3 & q_4 \end{pmatrix} \begin{pmatrix} b_0(1) \\ b_0(2) \end{pmatrix}$$

where $b_j(i)$ represent the i^{th} term of the b_j matrix. Performing the computation we get an expression for the first term like:

$$b_j Q b_0^t = b_j(1) b_0(1) q_1 + b_j() b_0(1) q_3 + b_j(1) b_0(2) q_2 + b_j(2) b_0(2) q_4$$

which can be brought in the form:

$$b_j Q b_0^t = \begin{pmatrix} b_j(1)b_0(1) & b_j(2)b_0(1) & b_j(1)b_0(2) & b_j(2)b_0(2) \end{pmatrix} \begin{pmatrix} q_1 \\ q_3 \\ q_2 \\ q_4 \end{pmatrix}$$

Vector $(q_1, q_2, q_3, q_4)^t$ is the result of the vectorization of matrix Q. We denote this with $Vec(Q)$. Thus we can express the left term of equation (B.1) as:

$$\sum_{i=j}^{n} \begin{pmatrix} b_i(1)b_{i-j}(1) & b_i(2)b_{i-j}(1) & b_i(1)b_{i-j}(2) & b_i(2)b_{i-j}(2) \end{pmatrix} Vec(Q) \tag{B.3}$$

Let's denote the row vector resulting from the sum with the letter Ω, so that (B.3) can be written in the form $\Omega Vec(Q)$.

Finally recalling the expression of the sample autocorrelation function we can substitute $E[\xi_k \xi_{k-j}^t]$ with:

$$E[\xi_k \xi_{k-j}^t] = \frac{1}{N} \sum_{k=1}^{N-j} \xi_{k+j} \xi_k^t$$

If we now denote with θ the following quantity:

$$\theta = \frac{1}{N} \sum_{k=1}^{N-j} \xi_{k+j} \xi_k^t - \sum_{i=j}^{n} a_i R a_{i-j}^t$$

we end up with an undetermined linear system in the form $\Omega Vec(Q) = \theta$ which solution is given by:

$$Vec(Q) = \Omega^{\dagger} \theta \tag{B.4}$$

where $\Omega^{\dagger}$ denotes the Penrose-Moore pseudoinverse of Ω.

Bibliography

[1] Francesco Cadini. Particle filtering for diagnostic, prognostic andpredictive maintenance. University Lecture, 2017.

[2] Leonid Sigal. *Human Pose Estimation*, pages 362–370. Springer US, Boston, MA, 2014.

[3] Broz Frank and Gordon Geoffrey. Better motion prediction for people-tracking. 03 2004.

[4] Matteo Ragaglia. *Towards a safe interaction between humans and industrial robots through perception algorithms and control strategies*. PhD thesis, Politecnico di Milano, 2016.

[5] Mao Ye, Qing Zhang, Liang Wang, Jiejie Zhu, Ruigang Yang, and Juergen Gall. *A Survey on Human Motion Analysis from Depth Data*, pages 149–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[6] S.J. Julier and J.K. Uhlmann. Reduced sigma point filters for propagation of means and covariances through nonlinear transformations, 02 2002.

[7] Cyrill Stachniss. Robot mapping: Unscented kalman filter. University Lecture.

[8] S. Kolås, B.A. Foss, and T.S. Schei. Constrained nonlinear state estimation based on the ukf approach. *Computers and Chemical Engineering*, 33(8):1386 – 1401, 2009.

[9] Xinguang Shao, Biao Huang, and Jong Min Lee. Constrained bayesian state estimation – a comparative study and a new particle filter based approach. *Journal of Process Control*, 20(2):143 – 157, 2010.

[10] Simo Särkkä. Unscented kalman filter and particle filtering. University Lecture, 2009.

[11] Sebastian Thrun; Wolfram Burgard; Dieter Fox. *Probabilistic Robotics*. The MIT Press, intelligent robotics and autonomous agents series edition, 2005.

[12] Dr. Christopher Sherlock Lisa Turner. An introduction to particle filtering. 2013.

[13] Joanna Zietkiewicz Piotr Kozierski, Marcin Lis. Resampling in particle filtering – comparison. 2013.

[14] Vijay John, Emanuele Trucco, and Spela Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530 – 1547, 2010.

[15] Alexandros Makris, Nikolaos Kyriazis, and Antonis A. Argyros. Hierarchical particle filtering for 3d hand tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[16] L. Van Gool M. Bray, E. Koller-Meier. Smart particle filtering for 3d hand tracking. 06 2004.

[17] A. Papandreou-Suppappola I. Kyriakides, D. Morrell. Multiple target tracking with constrained motion using particle filtering methods. 01 2006.

[18] Wikipedia. Kinect — wikipedia, the free encyclopedia, 2017. [Online; accessed 8-November-2017].

[19] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. E. Saddik. Evaluating and improving the depth accuracy of kinect for windows v2. *IEEE Sensors Journal*, 15(8):4275–4285, Aug 2015.

[20] K. Khoshelham. Accuracy Analysis of Kinect Depth Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3812:133–138, September 2011.

[21] Hairong Wang, Zhihong Deng, Bo Feng, Hongbin Ma, and Yuanqing Xia. An adaptive kalman filter estimating process noise covariance. *Neurocomputing*, 223(Supplement C):12 – 17, 2017.

[22] ZOUBIN GHAHRAMANI. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42, 2001.

[23] N.J. Gordon. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6), April 1993.