Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Automation and Control Engineering

Enhanced particle filter with environment reconstruction for human pose estimation in human-robot interaction

Supervisor:	Prof. Paolo Rocco
Co-supervisors:	Ing. Costanza Messeri

Master Thesis dissertation of:
Lorenzo Rebecchi Id. 893763

Academic year 2018-2019

Contents

List of Figures

Abstract

Collaborative robotics aims at integrating the accuracy and repeatability of an industrial robot with the skills and flexibility of a human operator in performing complex tasks, thus increasing the level of adaptability of the whole team. Robots designed to work side by side with humans often rely on vision sensors to acquire knowledge about the surrounding environment and their human partners, both for safety and for coordination issues.

Therefore, a potential improvement in this field consists in increasing the accuracy of the human pose estimation. This problem requires to determine in real time the space occupied by the operator, as well as his/her movements. This goal clashes with the difficulty of acquiring and correlating enough data while relying on a single fixed sensor.

The purpose of this thesis is to provide an innovative approach that improves the human pose estimation procedure, while reducing the associated uncertainty. To this aim, a constrained version of the Particle Filter algorithm was created. In fact, the traditional Particle Filter algorithm was integrated with a detailed virtual model of the real environment where a human operator and a robot can cooperate. Moreover, a constraint based on the limits of mobility of the human joints was added.

In this way the estimate of the unknown or uncertain human pose is allowed to propagate in a well-limited volume, whose boundaries are given by the geometrical surfaces of the operator's environment and the ones due to his/her physical structure.

The user's environment was reconstructed using the Mixed Reality device Microsoft HoloLens v1, while the human skeletal poses during the working activity were acquired with the Microsoft Kinect v2.

To build a unique model able to describe the workspace, these data were merged using a technique based on the simultaneous pose estimation of a fiducial marker, a ChArUco board.

The results obtained in a comparative experimental validation in a realistic industrial scenario showed that the improvements to the algorithm reduced both the error and the uncertainty associated to the estimation procedure, respectively of the 25% and of the 90% on average, with respect to the original Particle Filter version.

Sommario

La robotica collaborativa mira a integrare la precisione e ripetibilità dei robot industriali con l'abilità e la flessibilità di un operatore umano nell'eseguire compiti complessi, aumentando quindi il livello di adattabilità del team collaborativo. I robot progettati per lavorare fianco a fianco di utenti umani fanno spesso affidamento su sensori di visione per acquisire conoscenza circa l'ambiente circostante e i loro collaboratori, per esigenze sia di sicurezza che di coordinazione.

Di conseguenza un potenziale miglioramento in questo settore è l'incremento della precisione della stima della posa dell'operatore. Questo problema richiede di determinare in tempo reale lo spazio occupato dall'utente, così come i suoi movimenti. Tale obiettivo tuttavia si scontra con la difficoltà di ottenere dati sufficienti utilizzando un singolo sensore in posizione fissa.

Questa tesi mira a fornire un approccio innovativo che migliori il processo di stima e ne riduca l'incertezza. Per perseguire questo scopo è stata creata una versione di Particle Filter con dei vincoli aggiuntivi. Infatti l'algoritmo tradizionale del Particle Filter è stato integrato con un modello virtuale dettagliato dell'ambiente dove si trovano a collaborare l'operatore e il robot. E' stato inoltre aggiunto un vincolo basato sui limiti di mobilità delle articolazioni umane.

In questo modo ai possibili risultati della stima della posa dell'operatore in situazioni di incertezza è consentita la propagazione in un volume ben definito, i cui limiti sono dettati dalle superfici dell'ambiente di lavoro e dalla struttura fisica dell'utente.

La ricostruzione della cella di lavoro è stata effettuata utilizzando un dispositivo per la Realtà Mista, HoloLens v1 di Microsoft, mentre la posa dello scheletro dell'operatore durante l'attività lavorativa è stata acquisita utilizzando Kinect v2, sviluppato anch'esso da Microsoft.

Per creare un modello in grado di descrivere lo spazio di lavoro, i dati sono stati integrati tra loro utilizzando una tecnica basata sulla stima simultanea della posa di un marker fiduciario, una scheda ChArUco.

I risultati ottenuti in una validazione sperimentale comparativa in uno scenario industriale realistico mostrano che i miglioramenti apportati all'algoritmo riducono sia l'errore che l'incertezza associati al processo di stima, rispettivamente del 25% e del 90% in media, rispetto alla versione originale del Particle Filter.

Chapter 1

General Context

1.1 Introduction

Collaborative robotics is a paradigm of the so-called industry 4.0, where the collaborative robot (or cobot) is a smart artificial agent specifically designed to work with humans, while being intrinsically safe. To be able to work side by side with humans, cobots require the knowledge of the human motions as well as of the surrounding environment. To this purpose, a crucial role is played by vision sensors, in particular by RGB-D cameras, which are able to retrieve both colour and depth images that encode spatial information. Through these kinds of sensors, in fact, it is possible to give to the robot a description of the user pose in the working space.

However, as mentioned in [20], a lot of advancements in this field are linked to the possibilities of a robot to recognize the users and evaluate in real time his/her actions, needs or intentions. By integrating this kind of information in their programs, cobots in future will be able to interact with their human companions in a more and more humane-like way.

The first step in the process of recognition of the user is the human pose estimation, which is a difficult problem to address, since a lot of factors play a vital role in how the user is rendered in an image. In fact, unexpected movements, bad lighting conditions, partial occlusions of the human body, self articulation of his/her limbs and presence of complex geometries in the images are all issues which have to be solved to produce a suitable accurate estimate.

Particularly interesting is the case where at a certain moment an object prevents the complete acquisition of the human pose by interposing between the camera and the user. This phenomenon is called occlusion and is one of the main sources of uncertainty. In fact, without incoming data, the only way to infer the human pose is to rely on a filtering algorithm, able to estimate the possible poses from the knowledge acquired in previous time instants, when the user was properly tracked.

Since uncertainty in the estimation process is directly correlated to both safety and productivity, its reduction is of paramount importance. To pursue this goal, an enrichment of the data used in the estimation procedure might be the most viable way. To obtain new and different data a new branch of technology, the Mixed Reality, was included in the framework of the collaborative robotics.

In the last few years Mixed Reality devices began to take place in the industrial world, in particular as a new way to train and assist during assembly processes, as shown in [1]. In fact, the Mixed Reality devices allow the user to project in his/her field of view some additional virtual contents, called holograms, superimposed to the real environment that surrounds him/her. This provides a very intuitive, flexible and user-friendly way to guide the operator through his/her tasks.

Moreover, some of these MR devices are endowed with the capability of capturing information, in particular volumetric descriptions, related to the visualized real environment. To do that, they are generally equipped with a number of sensors, which are capable of gathering information from the user perspective.

Classical approaches to detect the user environment still consist in relying on one fixed sensor that cannot fully characterize the working environment. This leads to an unnecessary increased uncertainty.

The main contribution of this thesis, namely the integration of the Mixed Reality capabilities into the human pose estimation procedures realized through a particle filtering algorithm, aims to overcome this problem.

To exploit the huge amount of data derived from the integration of a vision sensor and of a Mixed Reality device, this work presents a way to merge information coming from them. The unified data are used to generate a more complete model that fully characterizes the operator's workspace. This enriched model is then integrated to the human pose estimation algorithm, increasing its accuracy.

1.2 State of the Art

In literature, several approaches to the human pose estimation problem can be found.

From a technological point of view, approaches based on RGB-D cameras, in particular those built on the Time of Flight technology such as Microsoft Kinect [4], are becoming more and more widespread. This is happening because, as presented in [10], from the data coming from this kind of sensor a number of processes are possible in order to perform the extraction of various features of the user, such as the estimation of the hands' pose or the 3D reconstruction of the face.

By applying an algorithm as the one presented in [26] it is possible to obtain from a RGB-D camera the complete pose of a human skeleton by comparing the data from the sensor with a database. This is the same functioning at the base of the human tracking capability of the Kinect sensor. Anyway, when the operator is not completely visible by the camera the pose of some of his/her articulations cannot be correctly extracted. This situation is called occlusion.

To overcome the problems due to occlusion several approaches are possible. In [9] multiple Kinects are used simultaneously to try to prevent occlusions from happening by mixing different points of views. However this leads to redundant informations and the need of an integrated architecture between the various devices. [16] tries to detect and filter out the occlusion during facial recognition processes.

Another popular approach is to use the pose of the user as the state of a process to be estimated. The proper choice of an algorithm able to filter the data acquired from the sensor is therefore mandatory to obtain a good estimated result. Traditionally, this kind of problem is approached using the Kalman Filter. However as outlined in [27] the linearity and gaussianity hypothesis are not suited for this problem and would require the introduction of either a complex model or a complex approach.

The other filtering algorithm often used in state estimation problems is the Particle Filter. The algorithm in [17] uses it to overcome the occlusions happening in a human tracking problem, where the whole pose of the user is approximated using just its position.

In [7] a comparative study between the Extended Kalman Filter and the Particle Filter with Bootstrap resampling is presented and showed the far superior performance of the latter in solving the estimation problem in a generic setup. In [5] a comparison between the same two techniques outlines the importance of including constraints in the Particle Filter algorithm to propagate. [15] presents a Particle Filtering algorithm able to address multiple object tracking simultaneously and [18] a mix between the Particle Filter and the Kalman Filter.

In [13] an approach to track each articulation of the user independently implementing a Constrained Particle Filter with Bootstrap Resampling in a collaborative robotics scenario is presented. In this algorithm are included constraints to take into account the human body size during the estimation process. Moreover, it includes a novel method to reduce the uncertainty coming from the occlusion, exploiting the spatial data coming from a Kinect sensor.

Since the results presented in [13] show that a Particle Filter including constraints is better performing than the Extended Kalman Filter, this algorithm has been considered to be the most suitable starting point for our work.

To further increase the capabilities of this algorithm, we aimed to reconstruct the workspace in which the cobot and the human operator are working using a Mixed Reality device. To validate our approach we considered the work performed in [3] and in [14] which evaluate respectively the capabilities of Microsoft HoloLens in mapping buildings and in localize itself in 3D models basing on fiducial markers identification.

1.3 Thesis purpose

The goal of this thesis is to reduce the uncertainty arising in the human pose estimation process (based on the data coming from a RGB-D camera) when the user is subject to occlusions. This task was addressed by acquiring data from multiple sensors, which include a fixed RGB-D camera, Microsoft Kinect v2, and a wearable Mixed Reality device, Microsoft HoloLens. Then, a sensor fusion technique is applied to relate the data coming from those devices, this allows us to estimate the unknown or uncertain occluded pose.

To pursue this goal, as suggested by [8], we decided to apply the Constrained Particle Filter technique. This method approximates recursively the posterior distribution of the unknown state variable through a finite set of discrete values also known as ”particles” to which a specific weight is associated. The weight of each sample represents the likelihood of the sensor measurement given the state of the particle.

We aimed at reducing the volume in which particles are allowed to propagate, to obtain a more accurate estimate. In fact the dimension of this volume is correlated to the uncertainty of the estimation.

To perform this reduction we applied two theoretical constraints: a particle (which represents a candidate for the position of the human articulation which is being tracked) shouldn't cross a physical object and it shouldn't violate the natural range of motion of the corresponding joint. In this way, particles are prevented from moving in places which are not realistically reachable by the articulations.

The concept at the basis of the implementation was the generation of a virtual model representing both the environment and the operator. To do that two procedures were designed:

- Environment Reconstruction: before starting the working cycle the workspace is scanned by the operator using HoloLens. A 3D model of the room is generated in real time and displayed to the user to allow correction of the errors on the fly. This procedure is designed to be as intuitive as possible, while not sacrificing the precision.

- Avateering: during the working cycle a virtualized model of the user is generated based on the data coming from the RGB-D camera present in the working space. This model is coherently positioned inside the reconstructed environment. In the human model, called *avatar*, the limits of the range of motion of the articulations are represented as virtual objects, those are used to constrain the particle motions.

The two models are then integrated with the Constrained Particle Filter in a 3D development platform. In the obtained virtual environment the particles are instantiated as virtual objects, which are subject to laws similar to physics' ones acting in the real world. Hence in this setup particles cannot cross virtual surfaces, just like in reality a ball cannot travel through a wall. In this way we obtained a method to constrain the particles to lie inside boundaries defined by the programmer. In this way they take the meaning of representing the positions that the user's limbs can occupy in reality, without propagating in unreachable places representing positions which are occupied, for example by a table or a by a shelf, or are reachable only with unnatural movements that violate the human body's limits.

1.4 Thesis achievements

The main achievements of this thesis are:

- The successful integration of HoloLens in a collaborative robotics situation and the fusion of its data to the ones coming from Kinect.
- The reduction of the volume occupied by the particles with respect to the original Constrained Particle Filter algorithm, presented in [13], of more than 90% on average. The reduction of this volume is an indicator of the reduction of the uncertainty of the result of the estimation procedure. Moreover, in a collaborative scenario reducing the uncertainty of the estimate means to avoid unnecessary reductions of the robot working speed.
- A reduction of more than the 25% on average in the estimation error, again with respect to the original Constrained Particle Filter algorithm.

1.5 Thesis structure

This thesis is organized as follows.

- Chapter 2 addresses the Human Pose Estimation issue, describing in detail the Particle Filter algorithm and the Skeletal Distance constraint. Here also the Occlusion Detection method, used to constrain the uncertainty under occlusion, is addressed.
- Chapter 3 presents the tools that were used to implement this work, both hardware and software. More specifically, the hardware and software instrumentation (Kinect v2, HoloLens v1 and Unity) used to develop this thesis are illusrated here.
- Chapter 4 aims at giving to the reader an overview of how the tridimensional model of the working environment is generated. In particular, the issues coming from the Time of Flight sensors are outlined and the Spatial Mapping procedure, used to produce a model of the environment, is presented.
- Chapter 5 deals with the techniques used to refer data coming from one sensor to the other one. In fact, since we use both a RGB-D camera and a Mixed Reality headset in our framework, we needed

a tailored way to be able to merge the data. The method, based on fiducial markers pose estimation, is described in detail in this Chapter.

- Chapter 6 shows the core of this thesis contribution, explaining the details behind the virtualization techniques adopted and how these are related to the Particle Filter algorithm. A Section is devoted to describe the Collision Detection procedure designed for this algorithm.
- In Chapter 7 the performance of the proposed algorithms are tested on a realistic use-case where a static and a dynamic occlusions are present.
- In Chapter 8 some conclusions about this work are offered.

Chapter 2

Human Pose Estimation: State of the Art

2.1 Introduction

In this Chapter the human pose estimation problem is presented. To perform this task an RGB-D camera takes data coming from the working environment where both a human operator and a collaborative robot are present. Data coming from the camera are communicated in real time to the robot to make it aware about the human state, provided that the operator is sufficiently visible. Our intention is to face the problems that arise when an occlusion happens.

In fact, in this situation an object interposes between the user and the camera. The main issue when this happens is the degradation of the performances of the team, since in case of lack of proper tracking the robot operates at decreased speed or is even stopped, to preserve the user safety.

The estimation of the human pose at a given time instant based on the estimate retrieved in the previous instants is generally addressed through filtering algorithms. Several approaches based on the Kalman Filter and other Bayesian estimators have been proposed in the literature. After accurate analyses, in this thesis we implemented a Constrained Particle Filter algorithm based on the work presented in [8]. In fact, the Particle Filter technique is an alternative methodology with respect to the well known extended or unscented Kalman filters, that turns out to be effective even in the case of non linear systems and non Gaussian distributions. Moreover, the formulation of the algorithm allows us to directly manage occlusion problems by limiting the uncertainty associated to the estimated pose, based on the depth map comparison, as will be further clarified in Section 2.2.2.

In this Chapter we will present the implementation of the Constrained Particle Filter applied to the tracking of human joints, detailing the choices made in design phase. Eventually, in Section 2.3 we will discuss the critical issues concerning this algorithm and the proposed solutions.

2.2 Human model

With "human model" we refer to a complete description of the whole human pose based on a bunch of numerical variables. In particular we are interested in determining at every time instant the positions of the operator's upper limbs which could be more subject to occlusion during the usual working activities, namely hands, elbows and shoulders.

To fully characterize their positions, two types of approaches are possible:

- Joint space: the human model representation in terms of joint variables is given in Figure 2.1. For each arm, four variables are needed, since there are 3 DOF in the shoulder and one in the elbow. Moreover, other degrees of freedom come from the torso's angle and from the position in the space of the human, which can be approximated as a unicycle model, hence adding other 3 DOF. In order to compute the position of a joint with respect to an arbitrary reference system the entire direct kinematics has to be solved, starting from the pose of the unicycle.

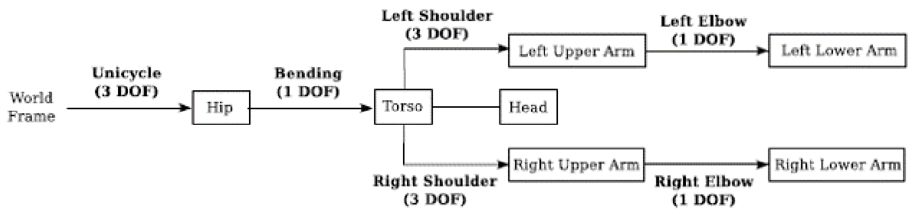

Figure 2.1: Human model schematics

- Cartesian space: each joint will be characterized by its Cartesian coordinates in world reference frame (x, y, z). These can be extracted from the data generated by the RGB-D camera using one of the approaches presented in literature, such as [25]. In order to retrieve the joint variables the inverse kinematic has to be solved.

The second approach was chosen in view of the fact that occlusions are easily imported and mapped in a Cartesian framework, whereas it's difficult to map them in joint space. Since we can estimate directly the position of the joints in this space the reconstruction of the pose of the user's legs and torso is not needed, thus we will restrict our analysis to the upper body.

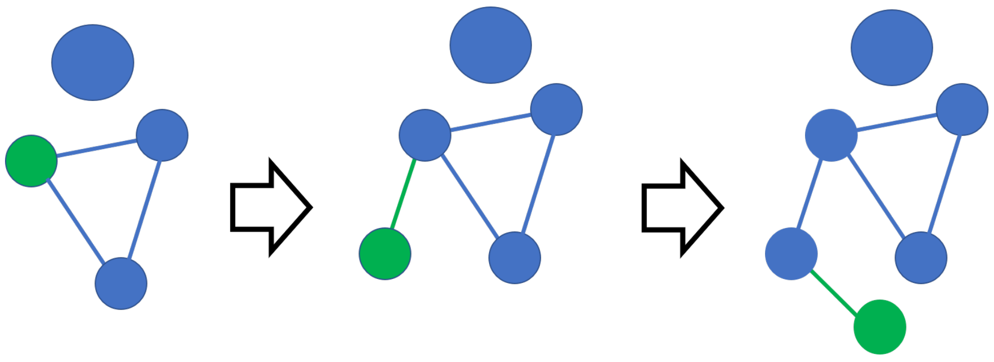
Figure 2.2: Joint hierarchy

Lastly, a hierarchy among joints is established, a visual hint of this procedure is given in Figure 2.2. In this, the order in which the articulations are considered and their position updated is set according to the direct kinematics: first the pose of the shoulder is estimated, then of the elbow and last of the wrist. In this way, when one joint's position is being estimated, it is possible to make use of the position of the previous one, as will be later discussed in Section 2.3.1.

2.3 Particle Filter

The expression Particle Filter refers to a class of non parametric algorithms used to infer the posterior distribution of the states of a Markov process. As shown in [23], it is used in a variety of situations including the Simultaneous Localization And Mapping (SLAM) problem, one of the hardest faced by robotics, and the tracking of moving objects.

The generic non-linear system to which this kind of algorithm is applied can be represented in state space as:

$$\begin{cases} x_{k+1} = f(x_k, u_k, w_k) \\ y_k = h(x_k, v_k) \end{cases}$$

where w_k is the process noise and v_k the output one.

The main difference with respect to the well-known Kalman approach and other parametric estimators is that the PF won't need a model of the system or even of the noise. In fact, the posterior distribution of the state is approximated through a set of samples (particles) which evolve independently from each other, used to simulate all the possible evolutions of the unknown state. This filtering technique allows good results even in case of non-linear systems and hidden states (as in the Hidden Markov Model).

Each of the N particles, indexed by i, is characterized by its state, represented as x_k^i, and a weight w_k^i, which is proportional to the probability that the measured state y_k is the same as the one of the particle. We denote with X_k the array containing the state of all the particles.

According to this notation, the posterior distribution can be empirically approximated as:

$$\hat{p}(x_k|y_k, \dots, y_1) = \sum_{n=1}^{N} w_k^i \delta(x_k - x_k^i)$$

where δ indicates the Dirac's delta function.

We will now briefly analyse the main steps of the Constrained Particle Filter algorithm which can be found in [8], to which the interested reader is referred for more details.

In this algorithms constraints which acts on the weight of the particles are added to modify the posterior distribution to better approximate the real evolution of the user's joints pose.

Some changes with respect to the literature were implemented in order to make the algorithm less expensive from a computational perspective.

The steps of the algorithm, represented also in a flowchart in Figure 2.3, for a single update phase are the following:

- *Initialization*: the first time in which the user is correctly detected and the target joint tracked N samples (particles) are drawn from the proposal distribution. The weight of each particle is set to $1/N$. In this phase the distance between the joint subject to the filtering and the previous one in the hierarchy is computed and stored.

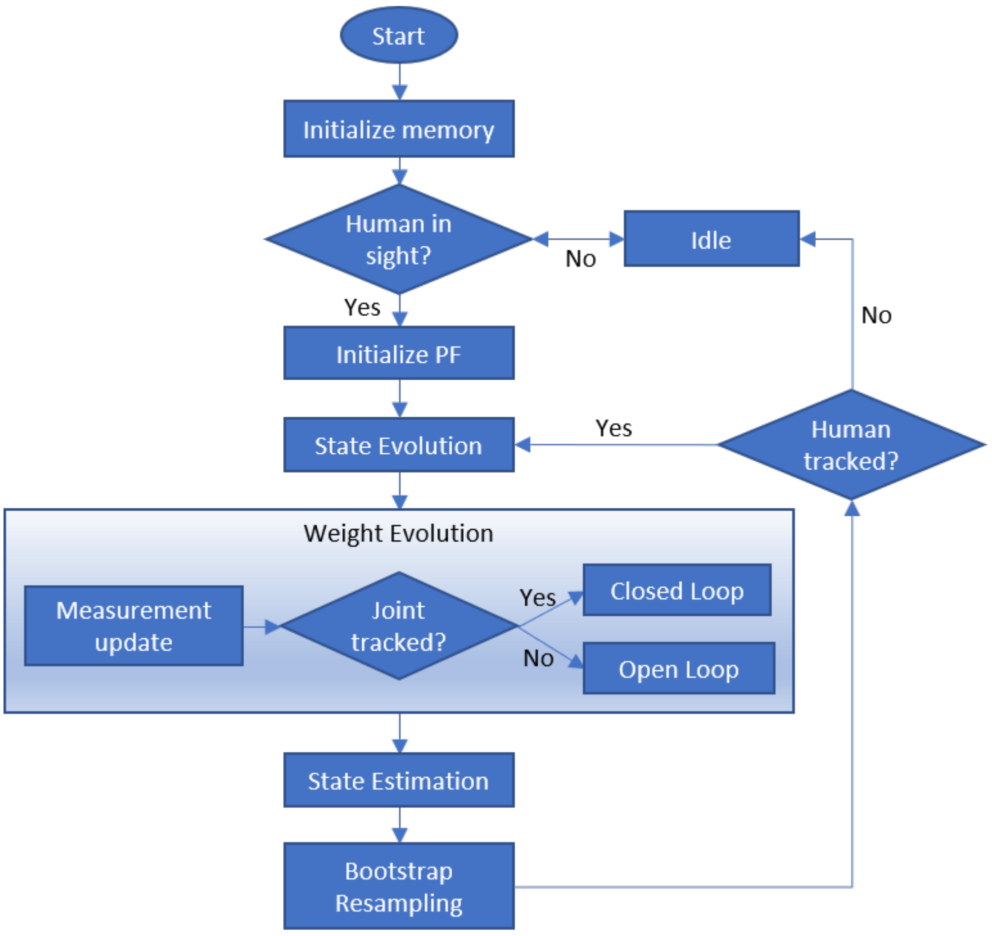

Figure 2.3: Flowchart representing the PF algorithm

- *State evolution*: the state of each particle stored in X_{k-1} is propagated according to the distribution $p(x_k|x_{k-1})$.
- *Measurement update*: if it is available, a new measurement of the joint position y_k is drawn from the sensor and the next phase runs in Closed Loop, otherwise it is called upon the Open Loop procedure.
- *Weight evolution*: In this phase the Skeletal Distance check is performed, as described in Section 2.2.1. Then, if a particle passes the test, two behaviours are possible according to the *Measurement update* outcome:
  - Closed Loop: the weight w_k^i is assigned to the i-th particle, proportionally to the likelihood of the observation given the sample:

    $$p(y_k|x_k^i) = \frac{e^{-\frac{1}{2}(y_k - x_k^i)R^{-1}(y_k - x_k^i)}}{\sqrt{2\pi \; det(R)}}$$

    where R is a diagonal matrix containing as elements the standard deviations of the noise acting on the measurements.
  - Open Loop: the particle is subject to the Occlusion Detection procedure which will be described in Section 2.2.2. If it survives, since there is no value for y_k, its weight is set equal to $1/N$.

  In both cases, the computed weights are stored both in W_k and in a cumulative weight array:

  $$W_{cumul}^k(i) = \sum_{j=0}^{i} w_k^j$$

- *Estimated state computation*: the estimated state for the joint is computed as:

  $$\hat{x_k} = \frac{1}{W_{cumul}^k(N)} \sum_{i=0}^{N} w_k^i \, x_k^i$$

- *Bootstrap Resampling*: particles are duplicated with probability proportional to their weight, this fundamental step is performed in order to avoid degeneracy problems [*]. At the end of this phase the new array of particles X_k is ready for a new iteration of the algorithm.

- For each particle a random number r is drawn from a uniform distribution in the interval $[0, 1]$
- The array of particle is scanned sequentially according to index i, when
$$\frac{W^k_{cumul}(i)}{W^k_{cumul}(N)} \geq r$$
the corresponding particle is duplicated and stored in X_k.

2.3.1 Motion model

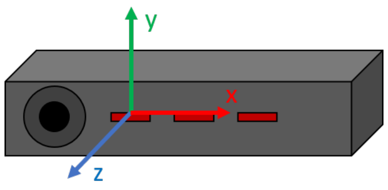

Figure 2.4: RGB-D camera reference frame

In this Section we will describe how the state of each particle evolves when the presented Constrained Particle Filter is applied. Due to the high computational cost of this algorithm, we will present its application to the problem of estimating the pose of a single joint.

In this formulation, the joint position and speed are expressed in Cartesian coordinates. This choice allows us to manage the occlusion in a better way and to avoid to compute the direct kinematics.

[*]The degeneracy phenomenon is a common issue related to the formulation of the generic Particle Filter algorithm. It happens when, after some iterations, all the particles except a few have negligible weight. Two methods exist to mitigate and overcome this issue:

- Good choice of the proposal distribution
- Use of resampling techniques

In this thesis the second option has been implemented using the Bootstrap Resampling approach.

The state can be thus expressed as $s = (x,\ y,\ z,\ \dot{x},\ \dot{y},\ \dot{z})$ where the coordinates of the particles are given with respect to camera reference frame, shown in Figure 2.4.

The Particle Filter is applied to the following discrete time system:

$$\begin{cases} s_{k+1} = As_k + \nu_k \\ y_k = Cs_k + \psi_k \end{cases}$$

$$A = \begin{bmatrix} I_{3x3} & I_{3x3} * \Delta t \\ 0 & I_{3x3} \end{bmatrix} \qquad C = \begin{bmatrix} I_{3x3} & 0_{3x3} \end{bmatrix}$$

Where ν_k and ψ_k represent the process noise, and output noise respectively.

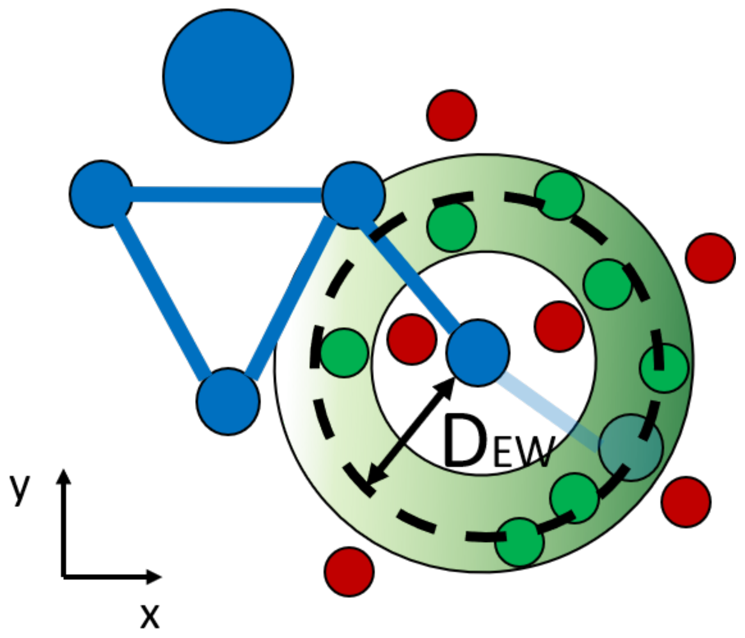

Figure 2.5: Bidimensional representation of the spherical crown in which particles are allowed according to the Skeletal Distance constraint. D_{EW} represents the skeletal distance between elbow and wrist. The intensity of the filler is intended to convey how is distributed the density of particles in this region.

To reduce the uncertainty of the estimate, to the standard formulation of the Particle Filter, as already mentioned, constrains are applied. The one limiting directly the possible values for the states of the particles is the Skeletal Distance test. This relies on the distance between two consecutive joints of the kinematic chain acquired during the *Initialization* phase, which is compared to the distances computed between the particles and the ”father” joint. If a particle violates the Skeletal Distance bound, this means that its position is too far or too close to the previous joint, hence its weight is set to zero and it is not subject to other computations in the *Weight evolution* phase of the algorithm. The resulting regions for particles are thus spherical crowns centred in the previous joint of the kinematic chain, as illustrated in Figure 2.5.

2.3.2 Occlusion Detection

As already stated, the expression occlusion is here used to denote an issue which prevents the acquisition of a joint position. This can be caused by:

- The presence of an object interposed between the camera and the target joint
- Noise acting on the scene, such as strong natural light

Since we are dealing with a controlled environment, we assume that the effects from the latter source of occlusions can be counteracted in setup phase. Therefore we will restrict our analysis to the first cause. To bound the uncertainties coming from this situation we implemented a second type of constraint, the Occlusion Detection procedure, which will be described in this Section.

When a lack of measurements due to occlusion forces the Constrained Particle Filter algorithm to run in Open Loop, the particles are propagated according to the Motion Model and constrained only by the Skeletal Distance bound, according to what was explained in Section 2.2.1. In this situation the possible positions of the particles generally exceed the region of the space hidden behind the object causing the occlusion, thus samples propagate in visible positions, this causes an unnecessary overestimation of the uncertainty of the pose.

Since we have at our disposal the data coming from the RGB-D camera, it is possible to determine whether a particle is in a visible position or not. This can be done comparing the coordinate Z of the particle with the depth map. A depth map is a particular type of image in which the pixel represents a value proportional to a distance, an example of it is given in Figure 2.9. This distance is the one between the camera and the objects in the scene measured along axis Z, which is normal to the camera plane.

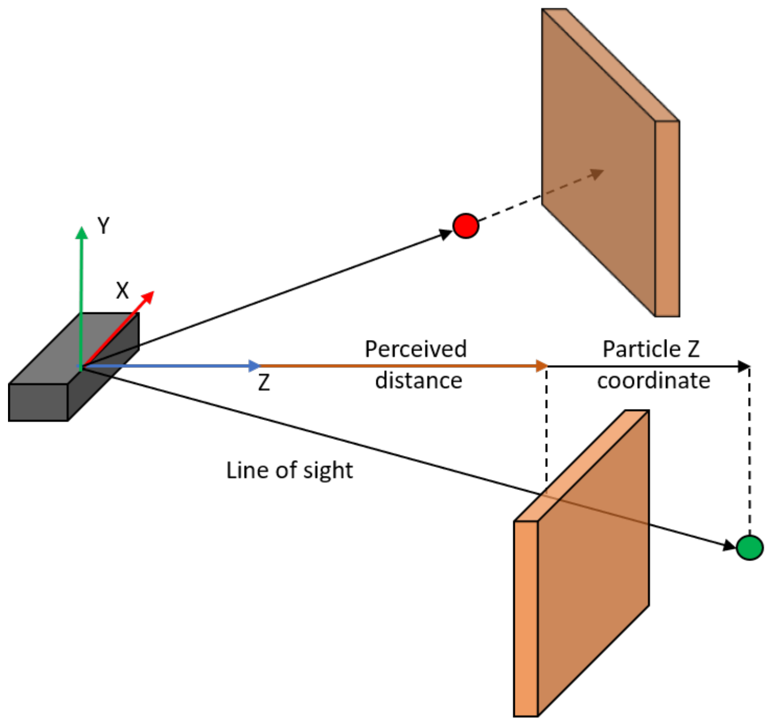

Figure 2.6: Visual example of the occlusion check

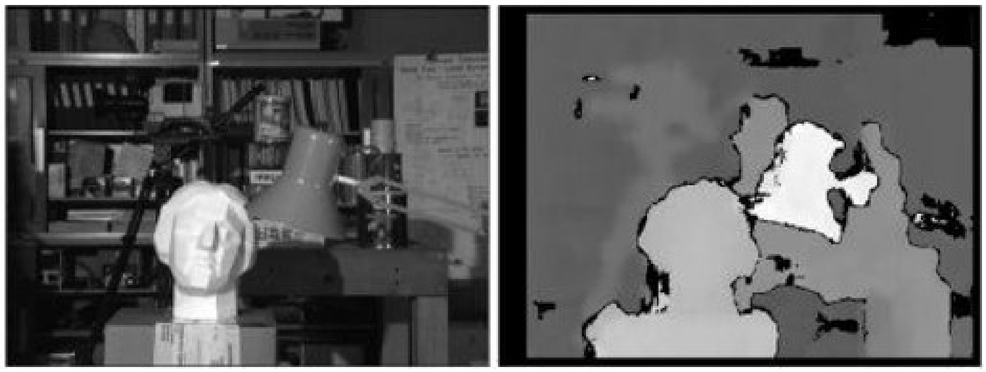

Figure 2.7: One of the two original image(left)and an example of depth map obtained using a stereo camera(right). The black regions are composed of pixels with value equal to zero.

The most straightforward approach to perform the distance comparison is to convert the coordinates of the particles (x, y, z) to the image ones (u, v) used to identify a pixel in the depth map. To do that, the following equations, based on the pinhole camera model, are used:

$$u = X * \frac{f_x}{Z} + c_x$$

$$v = Y * \frac{f_y}{Z} + c_y$$

where f_x, f_y are the focal lengths and c_x, c_y the principal point coordinates in the image plane. The resulting (u, v) are rounded to integer numbers. Then it is checked whether the corresponding pixel value is zero, which is a code used to indicate noisy detections or out of range distances. If it is zero, the closest non-zero pixel in the image is taken as reasonable approximation of the depth of that pixel: this procedure is called *neighbour filter*.

In the Constrained Particle Filter approach given in [13], if in a time instant the Occlusion Detection is required, all the pixels of the depth map are subject to the *neighbour filter* to eliminate the zero values. In our implementation we decided instead to save computational effort by applying it only to the pixels which are of interest for us, hence to those pixels whose coordinates (u, v) are obtained from the algorithm.

Eventually, as shown in Figure 2.6, the distance encoded in the pixel at coordinates (u, v), which is the distance perceived by the camera along the line connecting its focal centre to the particle position along the Z axis, is compared with the Z coordinate of the particle. If the first one is the greatest between the two, the particle represents an instance of the state in which the joint is not occluded. Since this hypothesis conflicts with the data coming from the camera, encoded in the depth map, it has to be rejected, hence the particle weight will be set to zero and it will be eliminated in resampling phase.

To sum up, when the joint is not tracked, all the particles that occupy positions which are visible from the camera are discarded, since if the joint had been in those places it would have been found.

2.4 Analysis and critical issues

The Constrained Particle Filter algorithm presented in this Chapter allows to estimate the human position in real time. However some critical issues were detected in its assumptions and implementations.

- The particles are allowed to propagate through objects, this can be misleading and increase the uncertainties of the estimation procedure. After an in-depth analysis, it has been found that this problem stems from the assumption of a working environment not known a priori and which cannot be explored in any way.

- The Motion Model for each particle doesn't consider the human joint limited range of motion. In fact, the only restriction to the particle possible movements comes from the Skeletal Distance constraint, which is satisfied provided that they lie on a spherical crown. This, anyway, allows the samples to move also outside the regions which are the natural range of motion for the human articulation. Therefore, when working near the bounds on joint variables, particles tend to propagate to unrealistic positions.

2.4.1 Proposed solutions

The main contribution of this thesis is to find suitable ways to improve the presented algorithm. In particular, a deep research has been carried out to analyse the most suitable and innovative approaches to increase the accuracy of the resulting estimate. The result, presented in the Chapters 3, 4 and 5, will exploit and integrate several aspects belonging both to the Mixed Reality field and the robotics to obtain solutions tailored on our needs.

In the end the solutions which will be implemented to solve the mentioned issues will be:

- Carrying out a preliminary exploration of the operator's working space using a Mixed Reality headset to reconstruct a geometrical model of the environment, where the user will be tracked by the RGB-D camera during his/her work, as explained in Chapter 3.
- Creating a tridimensional human model in a virtual environment using a 3D modelling engine. This operator's avatar will be placed in the generated room model, as detailed in Chapter 4. By using this merged modelling of the working scene it will be possible to:
  - Avoid particle propagation through the real world surfaces mapped previously.
  - Map the joint limits in Cartesian space as bounding volumes. By coupling them with the model movements we will further limit the space in which particles can propagate, thus reducing the uncertainty in the joint pose estimate.

Chapter 3

Setup

3.1 Introduction

The main goal of this thesis is to propose an advancement of the state of art techniques in estimating the pose of an operator when he is not completely visible by an RGB-D camera. In Chapter 2 we presented a Constrained Particle Filter algorithm suitable for this aim and analysed the main issues that tend to increase the uncertainty of the results.

The proposed method to reduce this uncertainty is to reconstruct both the user and the working space as integrated virtual models. To do that, a preliminary exploration phase of the room is performed using a Mixed Reality device, which generates a mesh describing the volumes surrounding the user. Then, during the working cycle, a RGB-D camera tracks the operator's pose, the data retrieved by this device are combined to those previously acquired and used to increase the capabilities of the Particle Filter algorithm.

Therefore, during this project we made use of two devices to acquire data describing both the environment and the user. Each of those sensors has its own methods, strengths and weaknesses. One of the main contributions of this thesis is to overcome their individual limitations by implementing a common framework based on a sensor fusion technique using fiducial markers recognition, that will be shown in Chapter 4.

In this Chapter we will present a discursive analysis of the devices and of the software components used to carry out the introduced procedures, with a short evaluation behind the pros and cons of their choice. The reader is referred to more specific works, such as [4], if interested in the technical details behind their functioning.

3.2 Sensors

The sources of data in our setup are two devices: a RGB-D camera and a Mixed Reality headset.

The camera is used in this work to recognize the human operator. Since we are interested in the skeletal pose of the user, which describes the position and orientation of each joint in the human skeleton, the choice fell on Microsoft Kinect v2. This camera is very accurate for the task required to it and embeds all the algorithms needed, but it is obviously a device designed to be kept fixed in the working space.

Since we are interested in mapping the environment in which the collaborative team works, we decided to exploit a Mixed Reality headset to perform the preliminary explorative phase. Those wearable assets add to the capabilities of a RGB-D camera all the sensors needed to keep an accurate track of their pose in the space. This feature eliminates the need to perform an odometric reconstruction based on the data acquired, which would be required by the Kinect or another RGB-D camera.

Moreover, the chosen device, Microsoft HoloLens, provides algorithms to convert the raw volumetric data into a more compact and usable form, the mesh, which will be exploited as explained in Chapter 6 when talking about the Enhanced Constrained Particle Filter. A mesh models the environment volumes using a series of basic geometrical shapes, such as triangles, to delimit the volumes.

Since we have data coming from two different devices using different formats, we adopted a common method to refer and integrate one to another based on fiducial markers. Those data are used to estimate the pose of a camera acquiring an image containing them, hence they introduce a common reference system with respect to which we are able to express the poses of the two sensors. The method itself will be further detailed in Chapter 4, however it is important to notice that the acquisition of a suitably accurate image for fiducial marker recognition using a movable camera such as HoloLens required a customized solutions.

In the remaining part of this Section we will present the specific devices used.

3.2.1 RGB-D sensor: Kinect v2.0

The Microsoft Kinect v2.0, shown in Figure 3.1, is a RGB-D camera and represents a standard in both the industrial and the research fields.

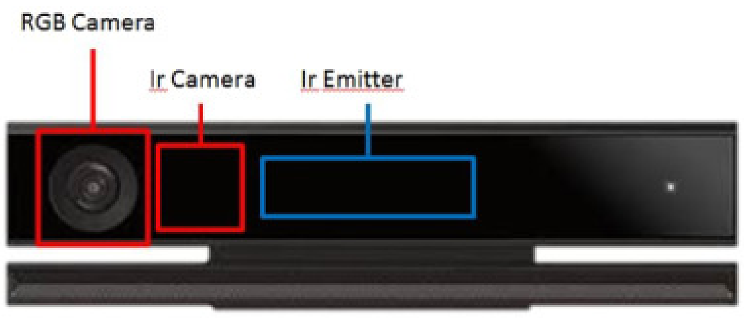

Figure 3.1: Kinect v2

It is able to acquire at the same time colour and depth images. From these, embedded algorithms retrieve skeletal pose of people in line of sight. Moreover, cheapness, extensive literature and availability of support scientific material concur to make it a perfect choice for this work.

It acquires colour images with an high resolution camera and the depth ones using a Continuous Wavelength Time of Flight sensor, whose characteristics will be discussed in Section 5.1.2. Besides, a clock is used to synchronize IR emitter and receiver to be able to acquire both an image where ambient light and infrared light are superimposed and one where only ambient light is present. Combining these two images it is possible to obtain also infrared pictures filtered from ambient light.

The spatial information given back by the this sensor is contained in a particular form of image, called *depth map* . This discretizes the portion of world that is seen by the camera into a matrix of pixels. Each pixel is characterized by its coordinates (u, v) in the image plane. The value of a pixel in the depth map generated by a ToF camera is assigned in two steps:

- The time that a light impulse spends travelling back and forth between the camera and the world is measured, from this measure it is possible to compute the distance along a specific line of sight.
- This distance is projected along the principal axis of the camera, this value is at last stored in the pixels.

In this way we obtain infos about the region of space in front of the camera.

A key feature for this thesis provided from the Kinect is that the associated libraries and software are capable of tracking and retrieving 3D positions of the skeleton of up to six people. The algorithm for acquiring

Figure 3.2: HoloLens v1

skeletal data is based on Machine Learning techniques. The output from these associates to each human in sight 25 joints, reproducing the poses of both the main articulations in the human skeleton and the limbs' ends. These data serve us as base of knowledge when running the Particle Filter, they are exploited during the *Measurement Update* phase.

3.2.2 MR headset: HoloLens v1.0

While the Kinect represents a well-known element in the scientific panorama, Microsoft HoloLens, shown in Figure 3.2, being released on 30th October 2016, is almost a novelty. This asset bases its Mixed Reality capabilities on a Universal Windows Platform architecture, a see-through display, and a number of sensors. Examples of usage of this device can vary from medical applications, such as in [29], to simulations of robotics and programming. Moreover, holographic feedbacks are recognized to be the most effective way to train people to assembly tasks.

This device is designed to be worn, the holographic see-through display allows the user to visualize virtual objects, known as holograms, displayed in a way able to reproduce their pose in the real environment. Those objects can be used to perform a variety of tasks, such as showing informations or provide an intuitive human-machine interface. Commands to the headset are given using both hands' gestures and voice inputs, while the cursor used to select items in the virtual world is moved using the gaze direction.

To further increase the realism of the Mixed Reality experience for the user, holograms are affected by the real world. For example, if an object interposes between them and the user they are not rendered, simulating their hidden position. Another key feature of this kind of device is the possibility of placing holograms in contact with the real world surfaces. To do that, these devices need a model of the volumes surrounding them.

This knowledge is acquired by HoloLens using the *Spatial Mapping*, that will be described in detail in Section 5.1.2. This technique is based on the simultaneous acquisition of spatial informations using the embedded Time of Flight camera and the tracking of the device pose in space. In fact, as in the Kinect case, HoloLens uses its Time of Flight sensor to acquire depth maps, which perspectives can be referred one to another knowing the pose of the device when they are taken. Since HoloLens mounts an Inertial Measurement Unit, a magnetometer and two stereo cameras, the localization of the headset is carried out with great precision. Therefore, the environment is viewed by the device as a series of bounding volumes, which are modelled using meshes.

One most favourable aspect which led to the choice of a this kind device instead of a simpler and cheaper sensor is that during the *Spatial Mapping* procedure the generated mesh can be visualized in real time as holograms, therefore any anomaly can be detected and corrected on the fly, even by a totally untrained operator.

3.3 Unity

In prototyping this thesis we needed a software able to manage and integrate 3D modelling, Kinect data extraction and Mixed Reality apps building. The most suitable environment for these tasks was the Unity engine, its logo is displayed in Figure 3.4. This software allowed to us to merge together techniques coming from very different fields.

Unity is a powerful 3D development platform which constitutes a standard for automotive, animation and research fields. As shown in Figure 3.3, it is able to render complex model of buildings and simulate various physics effects, such as collisions and gravity. Moreover, it is the recommanded engine through which a programmer can build Virtual and Mixed Reality apps. In fact, it is compatible with more than 20 different platforms, including HoloLens.

Unity provides a simulation environment where the programmer can design applications including virtual objects as well as their associated functionalities and test their behaviour even from a graphical perspective.

An app behaves like an object-oriented program, in which each item is an instance of a class defined by the programmer. Those instances are

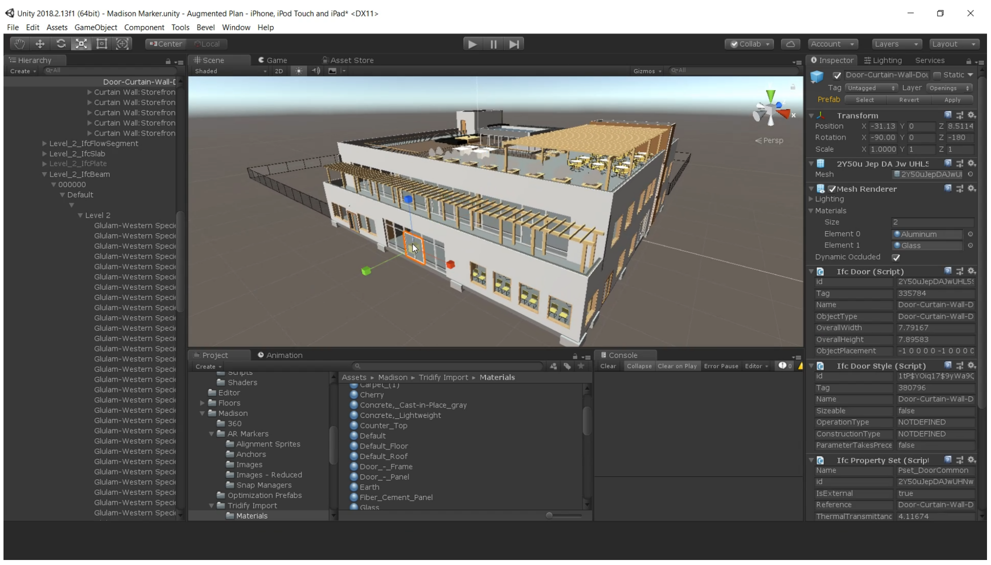
Unity 2018.2.13f1 (64bit) - Madison Marker.unity - Augmented Plan - iPhone, iPod Touch and iPad* <DX11>
File Edit Assets GameObject Component Tools Bevel Window Help
Hierarchy
Scene
Game
Asset Store
Inspector
Lighting
Services
Project
Animation
Console
Transform
Mesh Renderer
Ifc Door (Script)
Ifc Door Style (Script)
Ifc Property Set (Script)
Assets ▸ Madison ▸ Tridify Import ▸ Materials

Figure 3.3: Example of building modelling in Unity

Made with

unity

Figure 3.4: Logo displayed when an app is launched.

interactively affected by the final user at runtime and can be of several types depending on their function in the app's economy.

Tridimensional objects, such are holograms, in Unity are called GameObjects, are characterized by their pose in world coordinates, by a mesh renderer used to visualize them and by a collider, which is the feature describing the volume of an object and how it behaves when enters in contact with other entities. Moreover, several other items called Components can be attached to a GameObject to specify its features, giving the user an almost endless number of customisation options, in order to simulate every aspect of a real object.

The behaviour of the GameObjects is determined through scripts in C-sharp, developed and managed in the Visual Studio environment. These are used to define the logics in the apps and can access and modify the state of the entities. Programs are based on these scripts and the inputs from the user are processed by them.

The Physics module allows to simulate interactions between GameObjects, in particular to detect their eventual collision by computing the trajectories of a number of points on their collider. Several different approaches are possible to perform the management of collisions in Unity, our need for this type of techniques will be clarified in Chapter 6.

An app built in Unity is subdivided in scenes, each of them includes everything that is present in the environment, both GameObjects and scripts. Each scene has two subsequent initialization moments, used by the scripts by calling the methods $Awake()$ and $Start()$, and a potentially unlimited sequence of time steps, operations iterated over them are instantiated using the $Update()$ method.

To sum up, in Unity we found all the tools and techniques which were needed to get started in building applications simulating the evolution of the workspace in a 3D modelled environment, reconstructed a priori using HoloLens, in real time. Some of those tools proved more or less effective and in the end several different solutions were needed to overcome problems, particularly when dealing with the Physics module.

Chapter 4

Acquisition of the Volumetric Data

4.1 Introduction

In this Chapter we will present the procedures used to generate data that can replicate and describe in detail the geometrical shape of a generic room. This is done using HoloLens as a 3D scanner. As it will be explained in Chapter 6, the obtained tridimensional model will be used to improve the capabilities of the human pose estimation algorithm.

More specifically the aim of this Chapter is to present a way to represent the environment in which the user is located without having to rely on a CAD or another a priori produced model, which would generally require a lot of time and certain skill to be produced. Moreover, such model generally doesn't include movable objects normally present in the working space, whereas our goal is to include the maximum possible amount of details.

The procedure presented in this Chapter is an exploration of the environment, intended to be carried out once and offline and to be repeated only when a reconfiguration of the room is needed. Practically, the user scans the working space using HoloLens. The volumetric data obtained are preprocessed and results are presented in form of meshes, considered as the most compact and usable form for such information. Their poses are expressed and saved with respect to the world fixed reference frame of the Mixed Reality device.

4.2 Environment Reconstruction

This Section deals with the reconstruction of the bounding volumes in the environment based on the data acquired by the Mixed Reality headset. This device is worn by an operator during the scanning process. This procedure has been designed in order to be as intuitive as possible, requiring the human to just walk around slowly and scan the space with his gaze. The operation produces a series of meshes that are continuously updated and displayed in real time as holograms, as shown in Figure 4.1, in order to allow error checking. At the end of the process, a simple voice command allows the operator to store the outcome.

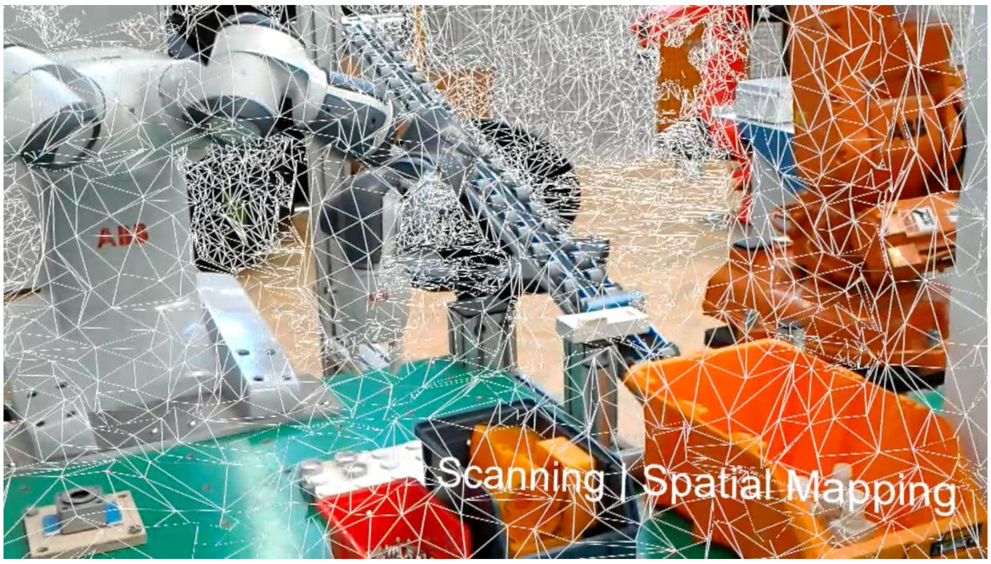

Figure 4.1: Frame taken using HoloLens during the Spatial Mapping procedure

HoloLens acquires volumetric data using a Time of Flight (ToF) sensor, which is the same technology used by the Kinect camera for retrieving data concerning the operator. This kind of sensor is used to obtain depth maps, but since it is prone to errors its functioning will be further detailed in Section 4.2.2.

The obtained depth map will be processed in real time by the Spatial Mapping in order to extract a mesh. Such meshes, composed of a series of basic polygons such as triangles or squares, have their coordinates fixed in the tridimensional space according to a global reference frame coinciding with the world frame of the Mixed Reality headset, which is generated at the beginning of a scene in Unity.

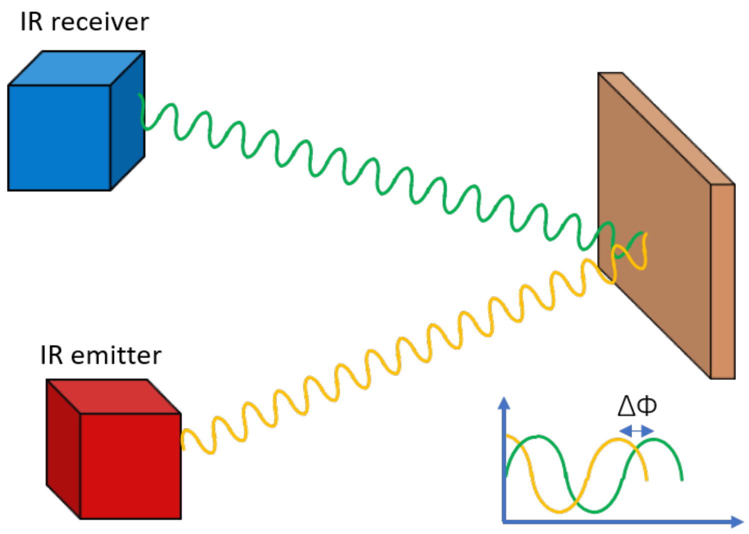

Figure 4.2: ToF Continuous Wavelength camera example

Since these technologies are the most recent developments in this field, in this thesis they have not been modified in any way with respect to their commercial counterpart, but they have been carefully selected and tuned in order to obtain the best possible result. The contribution of this phase will be clarified in Chapter 6, where the integration of the data coming from the Spatial Mapping with human tracking algorithm will be explained in detail. The rest of this section will be devoted to give an insight to the technical details behind these procedures.

4.2.1 Time of Flight sensors

Since both the environment and the human recognition accuracy will depend upon RGB-D cameras, this Section will present briefly the technology, namely Time of Flight-based cameras, used to acquire the depth component of the data. The analysis carried out is found on the informations provided in [24].

The name Time of Flight in computer vision refers to a technology used to estimate distances, based on the principle of measuring the time that a light impulse requires to start from an infrared (IR) emitter, bounce on the surroundings and come back to an IR receiver, as shown in Figure 4.2. This kind of camera was preferred to another popular technology to perform depth estimation, namely the stereo camera, due to the higher frame rate and precision, even though it is vulnerable to noise sources and the resolution of the generated depth images is generally lower.

In Continuous Wavelength approaches, the phase shift $\Delta\Phi$ to which the signal is subject when it is reflected is the main feature employed in computing the distance. Since this measurement depends upon the reflection capacity of the materials in the environment, it is worth pointing out some of the most likely causes of errors while estimating the depth:

- Concave objects: light impulses are sometimes reflected inside the cavity several times, this lengthens their time of flight and the perceived distance is therefore not properly estimated.

- Dark, transparent or highly reflective materials: material which will absorb too much or too little light can lead to misinterpreted positions.

- Too much natural light: since natural light include also frequencies which superimposes to those used by Kinect, this can increase greatly the noise on the estimation.

Moreover, since the computed distance depends over a periodical function which gives back results in the interval $[0, 2\pi]$ there is a bound on the maximum detectable distance. All these issues concur to force the use of these sensors indoor, in a controlled environment deprived of elements which could cause malfunctioning.

However, despite the efforts put in place, the factors which affect negatively the measures are too many. This is not an issue when estimating the human pose, since the Kinect can be placed accurately on the scene and kept fixed, but when it comes to room exploration using HoloLens the noise becomes relevant and can lead to misinterpreted geometries, which will in turn affect negatively the human pose estimation procedure.

4.2.2 Spatial Mapping

Applications in Augmented or Mixed Reality can benefit from information related to the environment surrounding the device. For instance, this data is useful to allow virtual objects, holograms, to interact with surfaces that have been recognised, such as walls, tables or chairs. Different techniques are available, depending also on the device in use, to map the surrounding environment, in this thesis the ones developed by Microsoft specifically for HoloLens are considered.

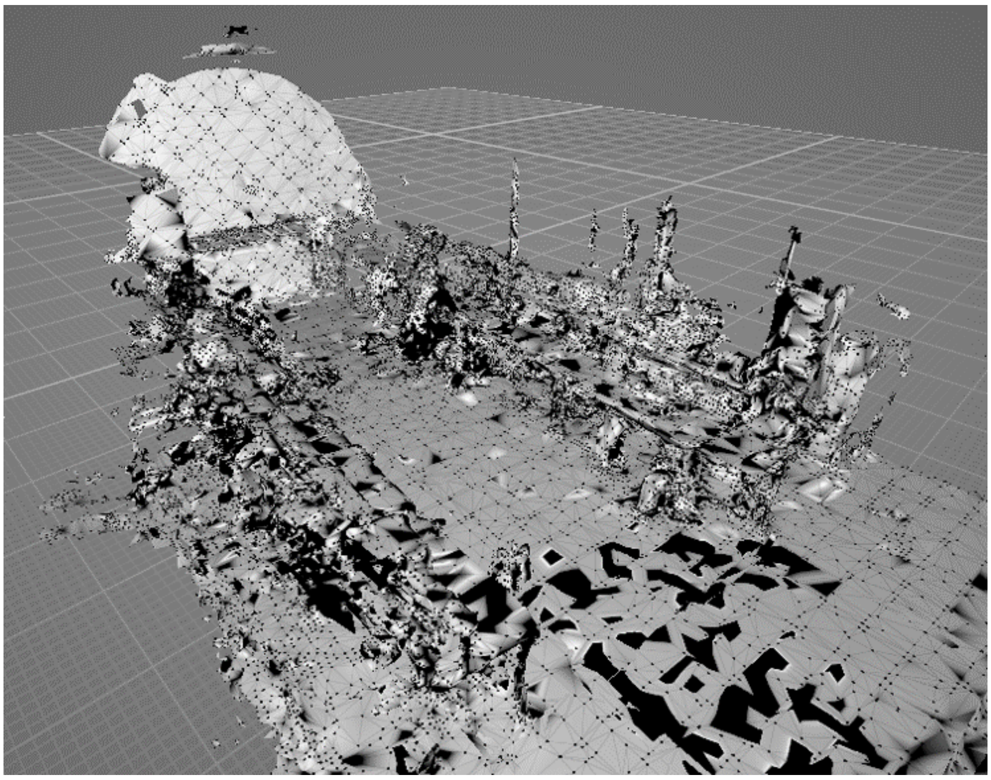

Figure 4.3: Example of mesh obtained from spatial mapping representing the laboratory

Spatial Mapping is a technology which aims to convert data coming from the HoloLens' depth camera into a more manageable format, called mesh, exploiting all the sensors available to the device to produce a complete model of the surrounding environment, as illustrated in Figure 4.3. However, this procedure is computationally expensive and tends to reduce the overall detail provided by the depth data generated by the Time of Flight sensor.

Our goal was to obtain from this process lightweight data representing continuous virtual surfaces (since our need for them is to constitute a boundary on which our virtual objects can collide) able to reproduce the geometry of the workspace. In this, the Spatial Mapping proved to be an invaluable asset, for the reasons that will be now be briefly described.

Generally, the depth maps acquired by a Time of Flight camera are used to generate point clouds. This format of data converts the information stored in the pixels, i.e. their coordinates on the image plane (u, v) and their values p, which are measures of distance from the camera, in a series of points whose Cartesian coordinates express exactly the same information as the original depth map, but can be referred to a reference system chosen by the user. Therefore, a point cloud is essentially a change of reference system for the 3D points described by the depth map's pixels, which instead of being expressed as a triplet (u, v, p) are expressed in the Cartesian standard (x, y, z) with respect to an arbitrary reference frame. Point clouds have the same number of points of the original depth map, thus are a heavy piece of information, and the space between them is void instead of a continuous surface, hence this format is not suitable for our needs.

The Spatial Mapping further elaborates the data stored in a point cloud to achieve the result that we were looking for. In fact, in a mesh the surfaces defining volumes are represented as a series of triangles, each one stored as three vertices to be read in a counter-clockwise order. To be more compact, adjacent polygons share one edge, thus needing to store just one additional point. Each triangle approximates a set of coplanar points in the point cloud, hence the mesh will need far less points to represent the same surface. Moreover, these surfaces are continuous, therefore a planar surface in reality is rendered as a plane composed of small triangles. Unfortunately, if a region of the depth map presents irregularities the corresponding mesh will be degraded, but less than the corresponding point cloud.

Since the Mixed Reality device is designed as wearable, it is subject to a localization procedure which updates at each time step a mobile reference frame attached to it. Position and orientation are estimated on the base of a number of sensors embedded in the headset, such as a Inertial Measurement Unit, a magnetometer and two depth-sensing stereocameras. Thanks to this feature, the generated meshes are positioned in the space according to a fixed reference system.

To further increase the accuracy of the output of this process the raw Spatial Mapping procedure can be improved by exploiting the Spatial Understanding asset. With this, during their acquisition, meshes undergo a process of object recognition aimed at discerning planes and ensuring they are represented correctly while reducing the noise, which can generate outlying vertices. In particular, this is a good way to ensure that the errors coming from the Time of Flight sensor are rejected. Furthermore, meshes are cropped and composed in order to avoid holes or overlapping, thus planes are precisely described.

On one hand, the usage of the Spatial Understanding capability has the advantage of rejecting a number of errors present in the depth map.

On the other hand, this procedure tends to reduce the details present in the standard Spatial Mapping meshes. For example, details such as small objects lying on a table tend to be not represented, and very often the concavities of furnitures such as shelves are filled up.

In the end, we decided to not apply the Spatial Understanding filter, since our goal is to map as accurately as possible the volumes with which the operator can interact, and therefore this filter would prevent us to render adequately boxes and shelves which could be present in the environment. The degradation to which planar surfaces (in particular the floor) are subject without the Spatial Understanding were considered acceptable in our setup.

However, we strongly recommand the usage of this powerful tool to map rooms and environments when not interested in the smaller details.

4.3 Implementation

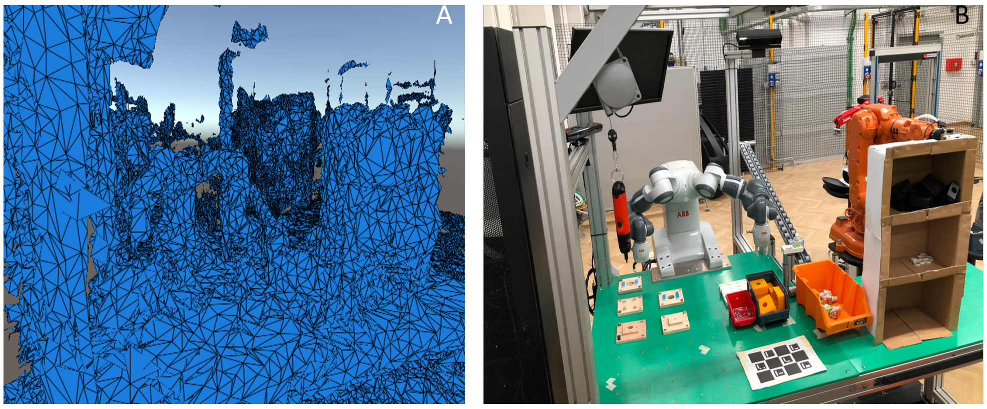

Figure 4.4: Example of mesh and environment compared

To apply the Spatial Mapping technique we implemented an app in Unity developed to run on HoloLens. This app has three key features: the capability of tracking the user's head pose in the workspace, the Spatial Mapping mesh acquisition and the real time displaying of the acquired mesh.

Regarding the first feature, the standard for Mixed Reality apps is to merge the tracking capabilities of the device with the simulation ones of Unity to superimpose virtual and real world in the user's vision. To do that the virtual world is mapped in the real one thanks to the sensors of the Mixed Reality device. The operator's point of view in the virtual world is simulated by Unity with a virtual camera represented in the scene. This determines what is shown to the user through the holographic display at each time instant. The position of the virtual camera in the virtual world is updated using the tracked movements of the real headset, in this way there is coherence between the real and virtual pose of the camera.

The implementation of the Spatial Mapping capabilities in our work was performed relying on the Mixed Reality ToolKit distributed by Microsoft. The app built in Unity includes in a single scene all the GameObjects needed to obtain the meshes and store them with coordinates expressed with respect to the fixed reference frame of the device.

The interface between the HoloLens' operative system managing the raw spatial data and the Spatial Mapping methods is constituted by a component called Surface Observer, which manages the low level extraction of meshes.

The meshes are then processed by the Spatial Mapping Observer according to parameters set up in design phase, to map a fixed volume of 5x5x5 meters with up to 2000 polygons for cubic meter. This element contains all the methods used to manage the meshes, store, update and discard them. Moreover, it determines the volumes in which the Surface Observer must work, in this way not all the data coming from the sensors have to be considered in real time.

The final result is shown to the user as a hologram representing the lattice of meshes, in this way he can evaluate if there is any region which has been mapped unsatisfactorily. Again, this reprojection of the result is based on the tracking capabilities of the Mixed Reality device.

When the process is terminated, a vocal command is used to store the mesh and close the app. The final result of the process is displayed in Figure 4.4, side by side with a picture of the workspace.

Chapter 5

Sensor Fusion

5.1 Introduction

The goal of this phase is to retrieve the homogeneous transformation matrix that allows us to relate the volumetric data coming from HoloLens to those acquired through the Kinect.

Since it is possible to extract the pose of a fiducial marker from a picture, the data correlation is practically done as follows: while an operator wearing the Mixed Reality headset scans a fiducial marker, the pose of the camera relative both to the one of the marker and to its own world reference frame will be estimated and stored. Meanwhile the same estimation is performed also for Kinect, already present on the scene in the same position that will be used later in human tracking.

Through an appropriate composition of the retrieved homogeneous transformation matrices it is possible to refer data coming from the two sensors to one another one.

5.2 Reference Systems Generation

In this Section we aim to realize the data transfer between different sensing devices. To do that, we exploited a fiducial marker and we placed it in a location that it is visible by both HoloLens and Kinect, as shown in Figure 5.1 from the perspective of Hololens. To obtain the transformation matrices needed to correlate the data coming from the devices, first the images taken by them have to undergo several passages to identify the marker, as presented in [22]. From the outcome of the identification

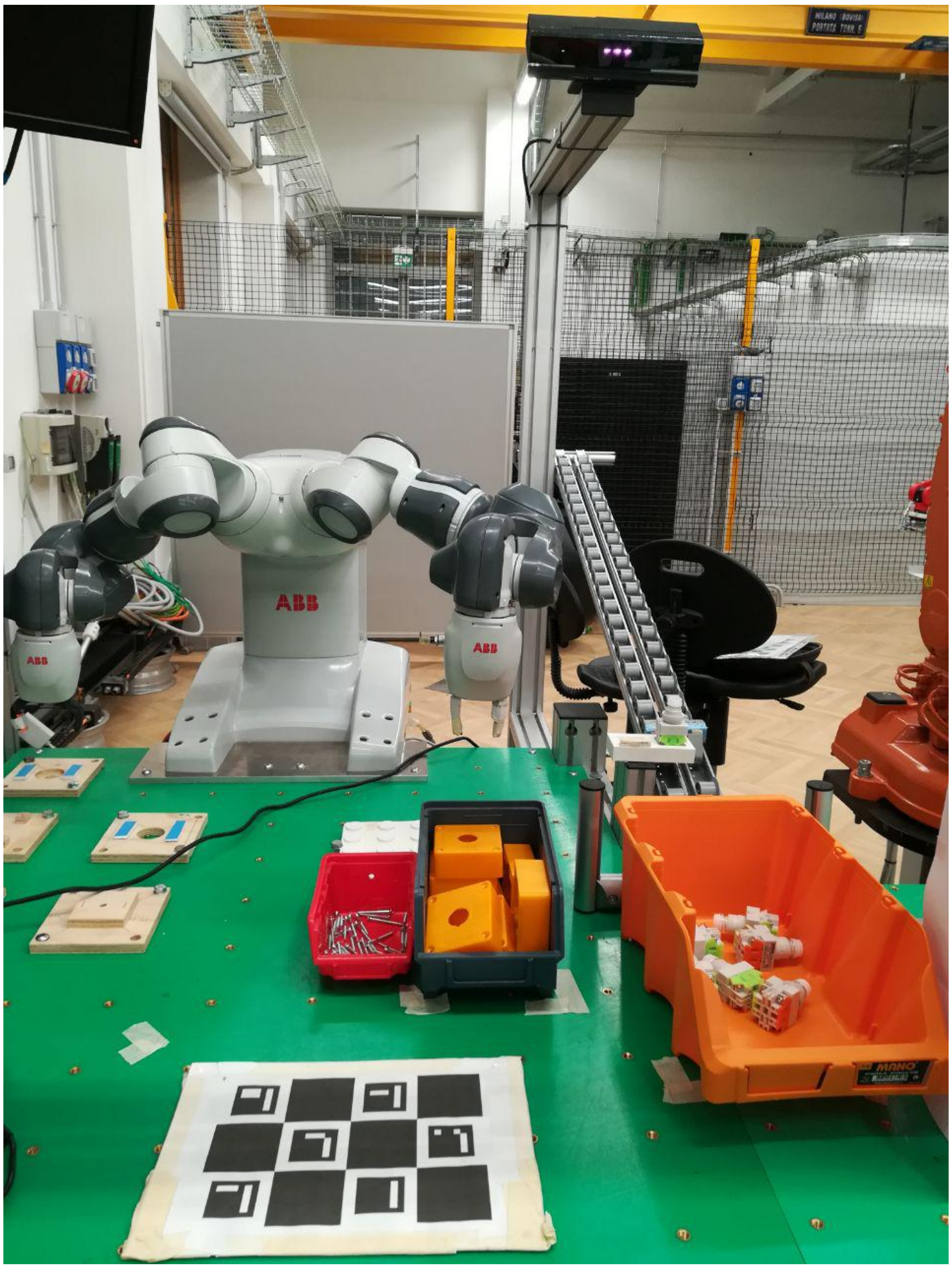

Figure 5.1: Positioning of the fiducial marker in the workspace, viewed from the point of view of the user wearing HoloLens.

phase, it is possible to derive the solution of the Perspective And Point (PnP) problem, which addresses the issue of estimating the pose of a

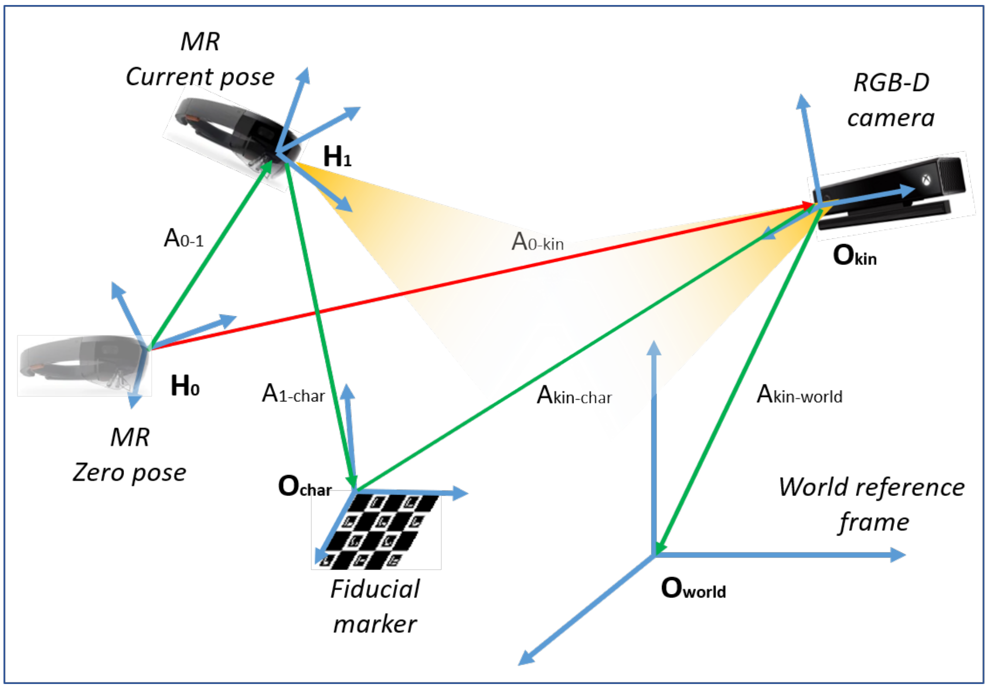

Figure 5.2: Overview of the reference systems present in the scene

calibrated camera given a set of coplanar points in Cartesian space and the corresponding projections on the image plane. The result of this solution is a transformation matrix expressing the pose of the marker with respect to the camera reference system.

In Figure 5.2 a graphical representation of our setup is provided. According to it, we define the origin of the reference system associated to HoloLens as H_1, the one associated to Kinect as O_{kin} and the last, associated to the fiducial marker, as O_{char}. The homogeneous transformation matrices between these three frames are obtained by estimating the marker pose, and we will denote them as $A_{kin,char}$ and $A_{1,char}$.

Since HoloLens is a mobile device, it refers its data to a fixed reference frame generated when the Mixed Reality applications are launched. This corresponds to the position of the user's head at the time instant when the app is launched. We called this H_0 and its pose with respect to the world reference frame is unknown to us. However, the Mixed Reality device has the embedded capability to track its pose changes with respect to H_0, in this way we can obtain $A_{0,1}$ and thus we can reconstruct H_0 pose.

To be able to compare data coming from Kinect and from HoloLens we will need to compose the matrices mentioned before. The composition needed is:

$$A_{0,kin} = A_{0,1}\, A_{1,char}\, A_{kin,char}^{-1}$$

It is worth to mention that the reference systems presented here have been simplified to make more clear this brief theoretical discussion. In the real implementation, since different software environments are needed by the devices to obtain the transformation matrices, the conventions adopted to express them are different and must be uniformed prior to the composition.

5.2.1 Fiducial markers: ChArUco

A fiducial marker is a (generally) planar figure with a predefined and known shape and dimension, which can be easily and uniquely identified using a camera. Examples of fiducial markers in the everyday context are the QR codes, used to encode tickets.

The standard techniques used in research and robotics to deal with the single camera localization problem are based on evaluating the deformation to which the image of the marker undergoes with respect to the normal view. From this knowledge the relative pose between the marker and the camera is computed. In this work a further step is taken, by using this technique to fuse together data coming from different sources estimating simultaneously the pose of the marker. This operation would require an high level of accuracy, in particular when dealing with a mobile camera.

ChArUco boards (see Figure 5.3) are a type of fiducial marker, which composes the best features from ArUco markers, presented in [2], often used in robotics, with the classical chessboard pattern. The pose of the chessboard can be better estimated thanks to the greater number of points available, corresponding to the internal vertices between black and white squares. Therefore, ChArUco boards were chosen among the other possible markers.

To work with this type of markers the *OpenCV* library was exploited. This is a well known open source library mainly used in Computer Vision and very intuitive in managing the images. Its integration with in the Mixed Reality app was one important contribution to this thesis, since customized ChArUco based functions have been implemented. To estimate the pose of a ChArUco board the following steps have been applied:

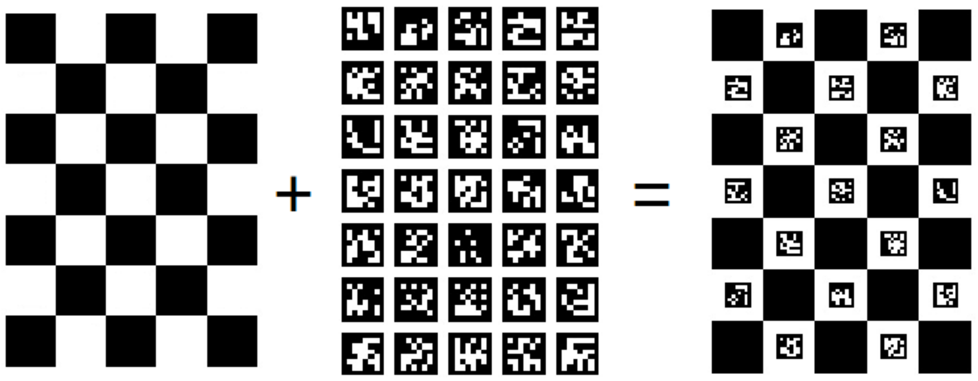

Figure 5.3: Composition of a ChArUco board, merging chessboard pattern and ArUco markers

- Identify the ArUco markers.
- Estimate the pose of each marker .
- Interpolate the positions of the chessboard corners.
- Estimate the pose of the board.

ArUco markers, one of the two components of a ChArUco board, see Figure 5.3, are square planar patterns constituted by a number of bits (also square) inside a black border. The infos associated to a marker are an id, which is generally an integer number encoded in its internal bits, its size in meters and two vectors relating the centre of the marker to the camera location. Four sets of coordinates representing the marker corners' positions with respect to the marker centre can be also easily obtained and converted between image coordinates and Cartesian ones, those are used to solve the pose estimation problem.

It is important to notice that the estimation of a single marker, namely of four coplanar points, is subject to ambiguity, particularly when the camera is not close to the marker. Since a ChArUco board requires two subsequent procedures for estimating its pose and the second one includes chessboard vertices (which are more precisely detected), the greater accuracy of the result lessens the risk of such errors. The main drawback is that, to obtain this better outcome, the images must be acquired at higher resolution and the camera have to be perfectly calibrated.

A point particularly critical in the fiducial marker estimation is represented by the need of ideal lighting conditions and a very little motion blur.

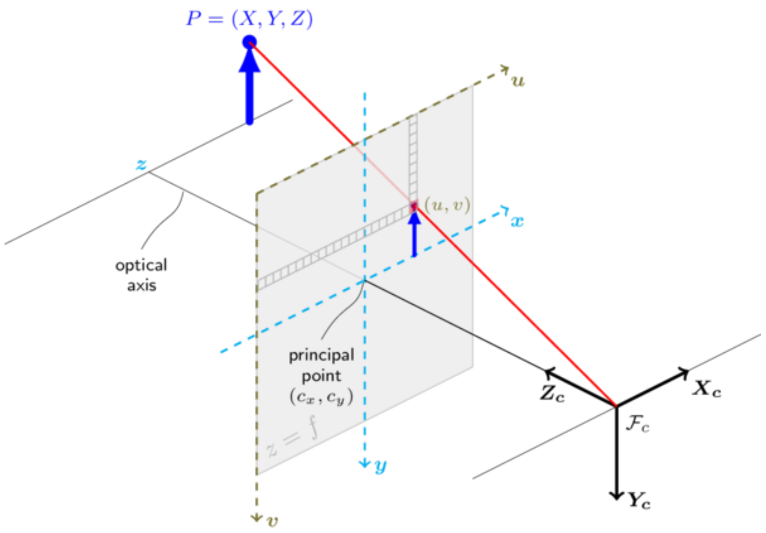

Figure 5.4: Projection of a point from Cartesian space into image plane. Image taken from OpenCV's documentation

These issues are solved in [6], proposing a different estimation procedure with respect to the classical one adopted in the OpenCV library. Anyway, due to the complexity of the solution mentioned in this reference, we relied on a simpler approach to reduce the magnitude of the errors, as will be explained in Section 5.3.

5.2.2 Perspective 'n' Point problem

To obtain the homogeneous transformation matrix relating the reference system of a calibrated camera and the one of the marker the Perspective 'n' Point problem has to be solved. This deals with computing the pose of the camera when the coordinates of a series of points are known both in Cartesian space and on the image plane, as shown in Figure 5.4. The output of this process is the matrix of extrinsic parameters describing the rotation and translation needed.

This problem can be formalized through the following equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Where (X, Y, Z) are the coordinates of a point in the Cartesian camera reference system and (u, v) the corresponding image reference system ones. The other matrices represent, from left to right, the camera matrix containing the intrinsic parameters estimated in calibration, a matrix necessary to uniform the dimensions and the matrix of extrinsic parameters, which contains all the quantities we are looking for.

Since an ArUco marker is composed of four points corresponding to its corners, whose coordinates are defined with respect to the centre of the marker, this problem is overdetermined, in view of the fact that solutions for classical approaches exist for a number of points greater or equal to 3. The reader is directed toward [26] for more details. The usage of a ChArUco board, due to the presence of an higher number of points in its structure, is supposed to give back better results provided that all the camera intrinsic parameters have been correctly computed.

Such parameters are used to describe the various deformations which an image undergo when captured by a camera. They have to be estimated through a calibration procedure, as explained in [19]. To obtain a properly high precision, several iterations have been necessary, particularly because of the movable nature of HoloLens. This device is prone to higher errors due to image blur and marker positioning at the border of the image, where distortion errors are greater.

In order to reduce the impact of these issues, the estimated pose of the marker is subject to an average over a number of frames, to avoid errors coming from a single distorted image.

Finally, since we needed an accurate calibration, we will now introduce the error function which describes the goodness of the calibration procedure, the so called reprojection error. This quantity corresponds to the error (in pixels) committed when trying to superimpose the projection of a previously detected point to its position in the image plane. It is generally considered a good calibration the one which gives back a reprojection error inside the interval [0, 1], measured in pixels. With our iterated process we reached projection errors around 0.5 pixels for HoloLens and of 0.2 pixels for Kinect, which we considered good enough. Again, the less accurate result obtained using HoloLens is grounded in its movable nature.

5.3 Implementation of the Mixed Reality ChArUco detection app

The app built for HoloLens in Unity implements the detection of ChArUco boards instead of a single ArUco marker *, to improve accuracy. However, the increased precision came at the price of needing a far more accurate camera calibration.

To import the OpenCV libraries into the Unity framework a customized dynamic-link library (dll) was built, wrapping the functions needed, such as *detectMarkers*, into code able to convert variables and data from C-sharp environment of Unity to the C++/C required from OpenCV. In this way we obtained the corresponding function that we called, for instance, *libraryDetectMarkers*. In this way the functions of OpenCV are made available to the Mixed Reality app.

The estimation of the board pose performed by those functions is done according to the what was described in Section 5.2.2.

At each time step the estimated pose is processed to reduce the noise on the estimate result. To do that, pose variations which are too little, due for example to the respiration of the user, are filtered out with an high-pass filter, and the resulting estimated pose is averaged over several time instant to reduce the effect of a single misjudged pose.

This last step is particularly important, since HoloLens is a movable camera. In fact, it is sufficient a slight motion blur or a bad light to prevent the marker detection or, worse, to generate a misjudged estimation. The latter is far more cumbersome, since it is difficult to detect and reject while processing data. Thus, the filtering of the incoming data results to be useful to reduce the impact of these sources of noise.

The resulting estimated pose is visually displayed to the operator by showing an hologram which is placed on the ChArUco reference system origin. Is is important to notice that to obtain a perfect alignment between the 3D coordinates of an object in the real world and the corresponding hologram projected on the see-through display, to that a user can see them perfectly superimposed from his point of view, a calibration procedure, known as SPAAM, should be carried out, in a way similar to the one shown in [21], otherwise a tolerance factor on the precision of the alignment needs to be taken in consideration.

*The ArUco detection app which was the starting point for our implementation can be found at: *https://github.com/KeyMaster-/HoloLensArucoTracking*

When the operator is satisfied of the result, a voice command is used to load the next scene.

Regarding the homogeneous matrices composition, the trivial issue of composing the matrices of rotation needed to refer the generated model to the Kinect reference system is complicated by the different conventions used. In fact, while Unity is left handed and uses quaternions to express rotations, Kinect and OpenCV are right handed, but the first uses quaternions and has its y-axis pointing upwards, whereas the latter uses axis-angle and the corresponding y-axis points down. These formats have to be uniformed before applying the transform to the room model obtained in Chapter 4.

Chapter 6

Enhanced Constrained Particle Filter

6.1 Introduction

In this Chapter we will present the improvements designed to increase the accuracy of the Constrained Particle Filter presented in Chapter 2. To do that, the volumetric data acquired through the Mixed Reality device during the Environment Reconstruction phase are used. These data will be imported in Unity, which was introduced in Chapter 3, together with the implementation of the Particle Filter and the assets needed to perform the human pose extraction from the data of the Kinect.

The presented technique was designed to improve the performances of the Constrained Particle Filter by further reducing the volume where the particles can propagate, thus reducing also the uncertainty related to the occluded human pose. Since the main structure of the algorithm is not changed with respect to the one presented in Chapter 2, here only the additional features will be detailed. These are concentrated in the *State Evolution* phase of the algorithm, since we are modifying the way in which particles can propagate.

6.2 Virtualized Particle Filter framework

In the Particle Filter the samples (particles) represent candidates to where the unknown or uncertain human pose joint could be located. For this reason we want to discard all those particles which violates physical constraints. In particular, we are interested in avoiding particles propagating through objects or outside the natural range of motion of the human joints. We call the regions of space which fulfil these boundaries "reachable volumes". To ensure that the particle remain in these volumes we started by importing the Particle Filter in Unity.

In Unity the model of the room generated in Chapter 4 can be imported and aligned with the Kinect reference system. In this way objects seen by the sensor can be placed in the corresponding position in the virtual world. In addition this simulated environment allows us to instantiate an articulated virtual model of a human detected by the camera. We will call the animated ensemble *avatar* and will be further detailed in Section 6.2.1.

In this virtual environment it is possible to instantiate each particle as a GameObject, which is the virtual equivalent of a physical object. In fact, they can move, collide and be visualized from the graphical simulator of Unity for debug purposes. Since our goal is to keep the particles inside the reachable volumes in the real world, we modelled the limits of these volumes with virtual surfaces, again instantiated in Unity as GameObjects.

This solution allowed us to convert the problem of confining particles to reachable volumes in the real world to the one of performing Collision Detection in the virtual one, since it is sufficient for us to avoid that particles travels through virtual surfaces to ensure that they fulfill the limits. We call Collision Detection the procedure used to determine whether two virtual objects touch or not.

At first we assumed that the Physics module of Unity would be sufficient to reach this goal. Anyway, to substantially improve performances, in the end we decided to implement our own Collision Detection technique, which will be later described in Section 6.3.

6.2.1 Human model

In this Section we will show the details behind the production of the human avatar used in our work as human model inside Unity.

First, it is important to remark that the choices discussed in Section 2.2, about the usage of Cartesian coordinates and joint hierarchy, are still applied here. We will then show the improvements derived from importing this model into a virtual tridimensional environment. Here, the user pose is modelled through a set of joints. An important difference

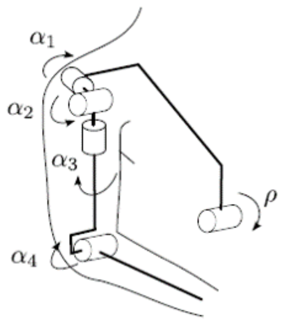

Figure 6.1: Human arm model and joint variables

between the approach shown in Chapter 2 and the one presented here is that, while producing the avatar, also the joints' orientations (which can be retrieved from Kinect) are considered.

In the following explanation we will refer to the human model with the same methodology that we normally use for a robot, as shown in Figure 6.1. To this aim, we recall the concept of kinematic chain for an arm as it was defined when introducing the joint hierarchy in Chapter 2: first comes the shoulder, then the elbow and last the wrist. The body part included between two articulations will be called link.

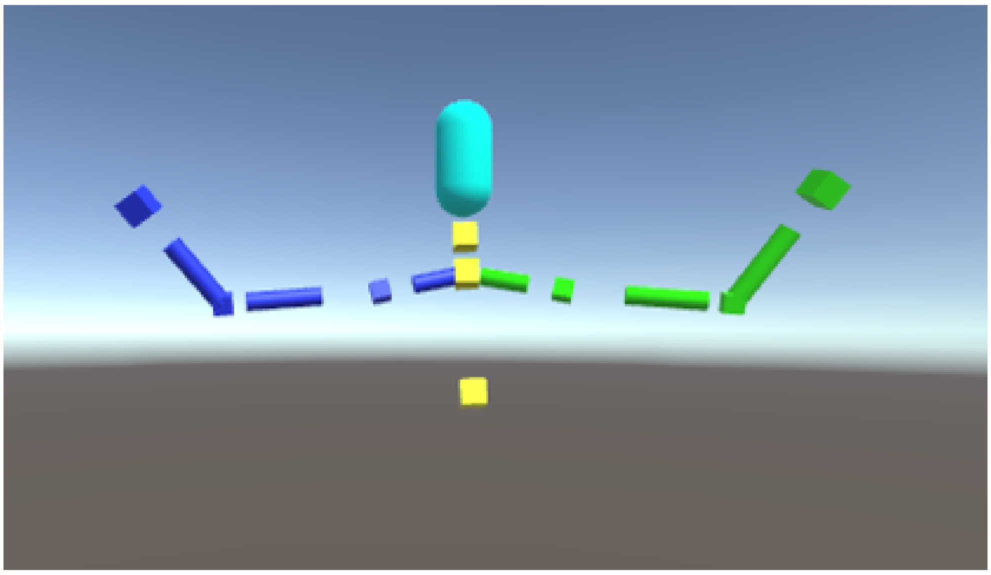

Figure 6.2: Human model in Unity, animated using the RGB-D camera.

We proceeded by associating to the pose of each joint in the real world the following objects in Unity:

- A GameObject representing the corresponding pose in the virtual environment. In Figure 6.2 those objects are the small cubes.
- A GameObject, the cylinders in Figure 6.2, representing the previous link in the kinematic chain. This means that the forearm will be associated to the wrist, the homer is associated to the elbow and something that is equivalent to the collarbone is associated to the shoulder. The length of the link is estimated based on the skeletal dimensions derived from the data coming from Kinect.
- The "Joint's Bounding Volume" for the particles which is used to estimate the position of the following joint, based on the joint variable limits.

The concept of Joint's Bounding Volume will be now detailed. Since we want to avoid the particles to model erroneous poses for the human body, we are interested in determining if they violates the physical range of motion of the joint to which they are associated. To do that we exploited the concept of kinematic chain, that allows us to proceed to the estimation of a specific joint relying on the knowledge of the position of the previous ones. In this way, for example, after that we computed the pose of the shoulder and of the elbow we know that the wrist should lie on a specific plane.

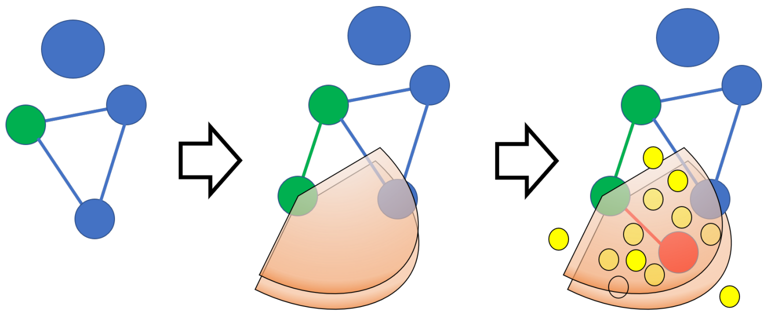

Figure 6.3: Sequence in the wrist Joint's Bounding Volume placement

The Joint's Bounding Volumes are modelled as GameObjects and positioned before the particles which they should constrain are propagated, as shown in Figure 6.3. The figure describes how, at each time step, the elbow position in Unity is updated after the shoulder, and simultaneously also the pose of the bounding volume for the wrist's particles is placed.

Then, the pose of the wrist is estimated, with or without the measurement availability. Since when the state of the particles is evolved the Joint's Bounding Volume is already in place, the particles will be restrained in their evolution inside it by the Collision Detection procedure described in Section 6.3.

The shape of the Joint's Bounding Volumes is obtained from the literature in [11] and they have to respect the following equations, according to the nomenclature of the joint variables described in Figure 6.1:

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

To implement these equations two strategies are possible: to keep the volumes shape fixed in a worst case scenario or to update the dimensions of the surfaces at each time step. The latter is obviously more tricky and requires to derive the joint variables from the orientations of the joints, which are known to us.

However, since we assumed in this thesis that occlusions are possible only for the wrist and focused our attention only on this joint, it is possible to see that once the orientation of the elbow is known the wrist pose depends only upon parameter α_4. Since the bounds on α_4 are independent from the other parameters, the two approaches mentioned before converge to the same one. Therefore it is sufficient to fix a volume according to the fourth equation.

The obtained volume for the wrist theoretically should be a plane. However this proved to be an hypothesis too strict, since when the motion of the wrist is perpendicular to the plane the number of particles respecting the bounding volume will decrease dramatically. Therefore the planar assumption was relaxed and the volume extended also in the third dimension.

The Skeletal Distance constraint introduced in Chapter 2 forces the particles associated to the wrist to lie on a spherical crown around the elbow, at a distance corresponding approximatively to the length of the forearm. Introducing also the Joint's Bounding Volume derived from the mentioned equations, the remaining propagation space is a 3D sector of arc, as shown in Figure 6.4.

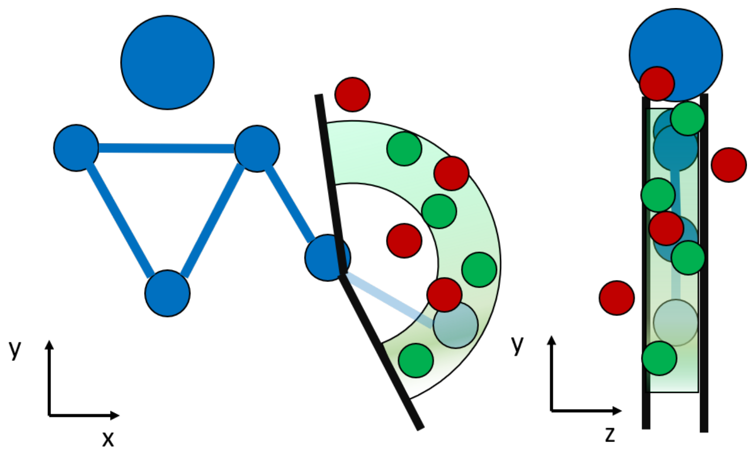

Figure 6.4: Joint's Bounding Volume example. Frontal view(left), lateral view (right)

This procedure represents our implementation of the limitations on joint variables and is not expensive from a computational point of view. In fact, the update of the pose of the bounding volume is performed simultaneously to the previous joint of the kinematic chain, the elbow, once at each time step. All the particles are then subject to the Collision Detection procedure, but this represents a fixed computational cost, as further explained in Section 6.3, thereforewe can conclude that there is no real loss of performance.

To animate the final model illustrated, in Figure 6.2, a technique known as *avateering* was applied. This is an innovative approach used to let an operator interact with a virtual environment directly with its body, bypassing the strict need of a physical interface with the machine. In fact, it couples the virtual avatar of the human to the skeletal poses retrieved from Kinect.

It is worth to briefly analyse here how Kinect extracts skeletal poses from depth maps, since we make an extensive use of those data. An example of the same kind of technique is given in [4]. In this paper a method (even more precise than the one natively included in the Kinect SDK) for estimating the human pose starting from a single depth map is proposed. It relies on the comparison between a point cloud generated from the depth map and a database of previously acquired meshes represented all the possible poses of a user. Once a correspondence is found, a refinement technique is applied to customize the skeleton to the particular body measures of the operator.

Hence, to sum up, Kinect estimates the skeleton pose as a whole and provides at each time step both position and orientation of each articulation of the skeleton. The frequency at which those data are refreshed is of 30 FPS, therefore it is not the bottleneck in our estimation process.

Considering this brief explanation, it is possible to say that to make the avatar reproduce the trajectories of movement executed by the user in real time two approaches can be adopted, similarly to what happens in robot programming: one in joint space and one in Cartesian space. We chose the Cartesian approach, since it is more straightforward and coherent with the strategies applied used for the rest of the model. However, since the orientations are also known, we exploited them to determine the orientation of the Joints' Bounding Volumes.

6.3 Collision Detection

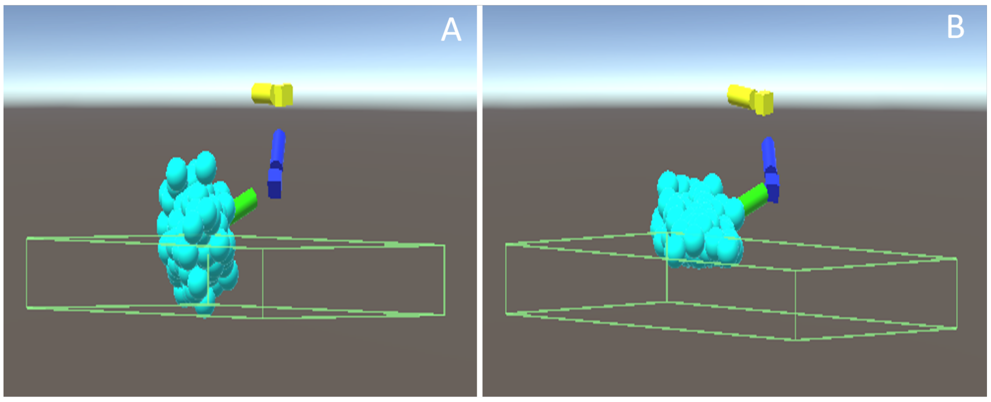

Figure 6.5: Particle propagation through a volume without and with Collision Detection in Unity

In this Section we will describe the main improvement applied to the Particle Filter algorithm presented in Chapter 2. This is the Collision Detection capability, which is a method introduced to determine how the particles represented in Unity as virtual objects interact with the other GameObjects. This is a key feature of our work, since on this interaction is based our capability to limit the particles' movements.

An example of the applicability of this procedure is illustrated in Figure 6.5. In this picture, with the Unity standard that we used through this entire work, the particles are rendered as light blue spheres, the user forearm and wrist as green objects, the elbow and homerus are rendered with dark blue colour and the shoulder and collarbone with yellow. The green lattice represents a planar surface (in this example a table) that we modelled in Unity with a simple tridimensional volume in a position correspondent to the one that it has in the real world. Therefore, we can say that the green lattice corresponds to a real world surface on which the user is placing his hand.

In the figure marked with "A", where the Collision Detection is not performed, when the operator moves his wrist close to the table the particles are able to move through it. On the other hand, the picture marked "B" shows the result with the Collision Detection enabled, which prevents the particles to propagate through the object.

The Collision Detection test is performed in the *State Evolution* phase, as can be seen in Figure 6.6, and is designed to prevent particles to cross surfaces present in the workspace modelled in Unity, which can be part of the mesh reconstructing the room, of the Joints' Bounding Volumes or of any other boundary of interest. In view of that, the issues outlined in Section 2.4 are solved.

To perform this detection we initially relied on the Physics engine already present in Unity, which was supposed to be able to manage the high number of objects moving on the scene. This engine tries to recognize when the GameObject representing a particle is in contact with a surface at discrete time steps. Anyway, if an object travels through another completely in the time between two updates, the collision is not detected, preventing our algorithm to work properly.

By enabling the suitable options, it is possible to force Unity to interpolate the trajectories travelled by the objects. Unfortunately, since this interpolation is performed over the whole collider (which is a lattice of points distributed on the outer surface of the GameObject) associated to each particle, the process is computationally very heavy, resulting in an excessive slow down of the execution. Moreover, it often happens that to preserve an appearance of real time performance Unity performs the trajectory interpolation only over a part of the particles. This results in a situation where a number of particles are not checked and are able to cross the surfaces which should contain them.

To sum up, since the standard built-in assets are not satisfactory, a workaround to solve this problem was designed by exploiting the raycasting technique. The raycast is a built-in method in Unity used to verify if

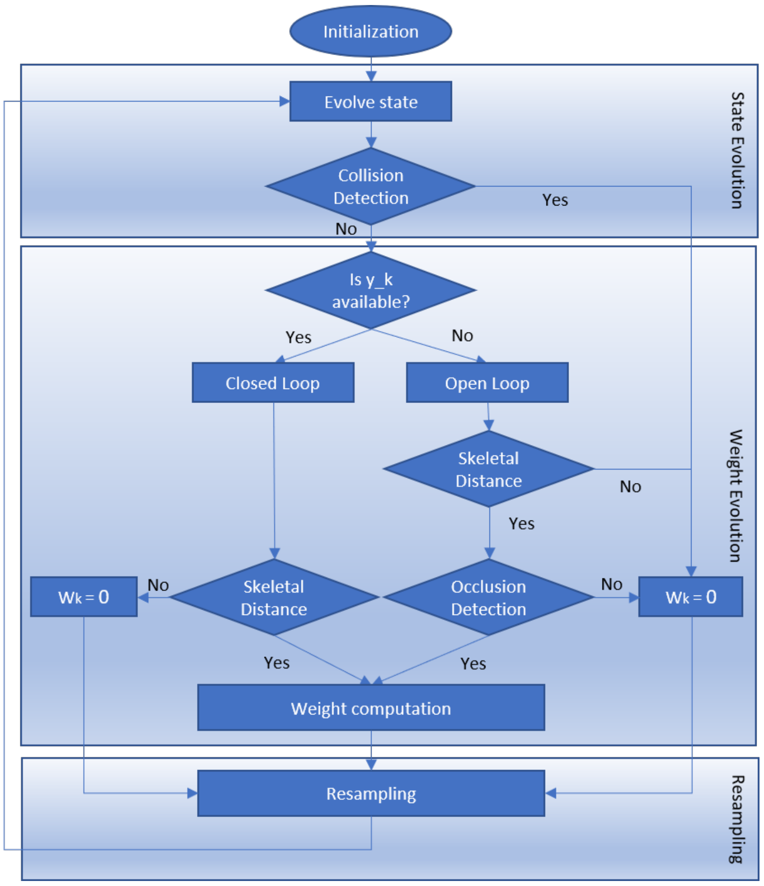

Figure 6.6: process to which a particle is subject at each time step

and where a collision happens along a line connecting two points in a 3D environment. It projects a line, called ray, along a specific direction from a starting point and gives as feedback the point in which it touches the first object. If the ray doesn't encounter any obstacle on its way, or reaches its maximum length, specified by the programmer, it returns a boolean "false" value.

The Collision Detection test based on raycasting takes place during the *State Evolution*, when the new state x_k is computed starting from x_{k-1}. When a new position for the particle is computed, a ray is cast from the elbow position at time k to the wrist position proposed by the particle, which is x_k. If the ray "impacts" a virtual object, the particle is violating one or more boundaries, hence its position is set to the one of the impact point, its weight is set to zero and we will say that it is "colliding". Moreover, a flag is enabled, which will prevent the particle to be subject to any other check and to be considered during the *Weight Evolution* phase.

By propagating the ray from elbow to wrist we ensure that neither the proposed pose for the wrist nor for the forearm are in violation of any physical boundary, provided that they are mapped inside Unity using the Spatial Mapping technique shown in Chapter 4.

The Collision Detection technique based on raycast proved to be fast enough, in fact around 500 rays increase the computational time of 1 ms. Moreover, since this procedure is carried out at each time step, it generates a fixed loss of FPS, whereas the standard Physics module would slow down when more collisions are detected.

6.3.1 Order of the particle tests

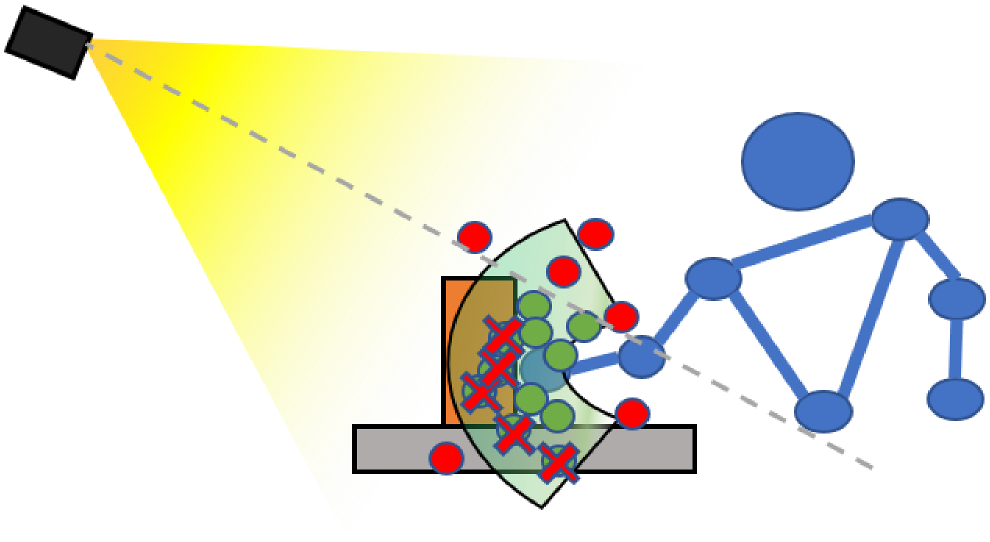

Figure 6.7: Example of particle check result in case of occlusion. The spherical crown allowed by skeletal distances is cropped to a section of an arc by the joint variables limitations. Particles lying in positions visible by the camera are discarded. Particles eliminated due to collision are outlined by a red cross.

The tests to establish if the position of a particle is valid are carried out following a specific order. If one check is failed the particle is discarded, its weight is set to zero and it will be deleted in phase of resampling. It is important to notice that the discarded particles are not subject to any other procedure, this is done to avoid wasting time. In this way we tried to reduce at a minimum the executions of the Occlusion Detection test, which is the most complex procedure and can slow down the application.

The order is:

- Collision Detection: explained in detail in Section 6.3, it is used to enforce the joint variable limits and to prevent the particles to cross the surfaces which were reconstructed in the virtual world as discussed in Chapter 4.
- Skeletal Distances: checks if the particles lie at a suitable distance with respect to the previous joint. It is exactly the same procedure explained in Section 2.3.1.
- Occlusion Detection: it is carried out only when an occlusion occurs and the algorithm is running in Open Loop. With respect to the literature it relies on a lighter implementation of the *neighbour filter* to fill up the holes in the depth map, applying it only to those pixels which are tested, as shown in Section 2.3.2.

The final result of all these procedures is that the volume where the particles are allowed to propagate is strictly reduced, using all possible knowledge about the human and the working environment. A visual example of this fact is given in Figure 6.7. It is important to notice that the Occlusion Detection is applied only when the algorithm is running in Open Loop and is based on data coming from the RGB-D camera. The reason behind this choice, instead of relying on a virtual simulation where particles are visible, lays in the fact that it helps managing the case when an occlusion not mapped a priori in the "Environment Reconstruction" phase happens.

6.4 Implementation

To implement the Enhanced Constrained Particle Filter we used Unity, which provided the framework to build the app integrating 3D models of the workspace built with HoloLens and data coming from the Kinect, which are used to model the operator. In Figure 6.8 are illustrated the data fluxes and the sequence of procedures performed before arriving at the Particle Filter application, with a distinction between the upper part of the graph, which contains all the processes performed offline with respect to the main working cycle, and the lower part of the figure, where the online procedures, which are the focus of this Chapter, are contained.

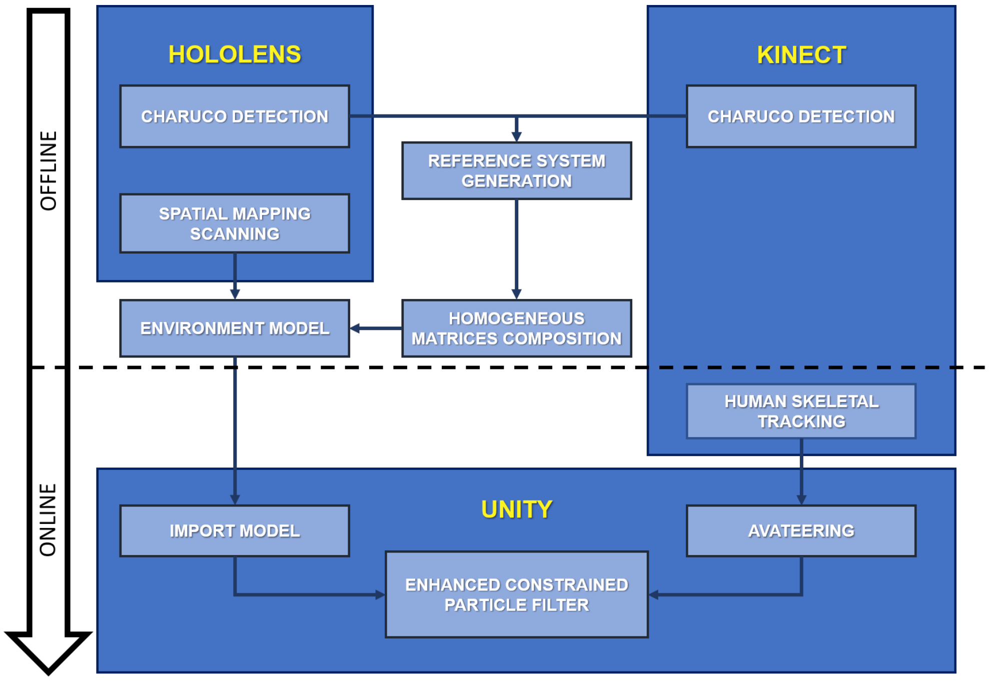

Figure 6.8: Process graph

In Unity, an interface for Kinect was created using the specific library KinectForWindows. This interface is constituted by two classes, one managing the body data acquisition, called Body Source Manager, and one responsible of retrieving and processing the depth map, named Depth Source Manager. At each time step, the Body Source Manager provides a set of data containing the tracking state for each joint and its complete pose. This pose is converted to be compatible with the reference system of Unity, which is left-handed-based whereas the Kinect works by relying on right-handed frame. At each joint is coupled a GameObject, according to the avatar described in Section 6.2.2.

Before launching the pose estimation algorithm, the room model obtained through the Spatial Mapping must be imported in Unity. This model is stored with a file containing the composition of the homogeneous transformation matrices used to align the room model to the Kinect reference system. The meshes uploaded in Unity are automatically converted in GameObjects. The pose of these objects is then modified applying the homogeneous transformation matrix computed before, converted in the Unity standard, which is to use quaternions to express rotations.

The Particle Filter is implemented in Unity as a class containing as attributes the vector of particles and all the support variables needed to run the algorithm. The Particle Filter is associated with a GameObject whose position coincides with the estimated state of the wrist.

When a human comes in sight, the Particle Filter initializes the particles creating a set of instances of the Particle GameObject around the first valid position in which the joint is tracked by Kinect.

The Particle class associated to the homonymous GameObject is responsible of managing at each time step all the procedures which involve a single particle. Those are the weight computation, the state evolution, the Skeletal Distance test and the Collision Detection.

The operations implying knowledge of all the particles, which are the state estimation and the resampling, are performed by the Particle Filter directly, accessing the state of the particles when needed.

During the State Evolution phase, as explained in Section 6.3.2, the Collision Detection is performed. The Skeletal Distance test is executed in the Weight Computation phase. At each time step the Particle Filter calls the method of the Particle class for computing the weight in Closed Loop or the one in Open Loop, depending on the joint tracking state.

The Occlusion Detection test is performed by a method of the Depth Source Manager class, which contains the depth map as an attribute. A preliminary step to this test is to convert the pose of a particle in the depth image space using the Coordinate Mapper from the Kinect library. We used this built-in function to avoid to have to calibrate the depth camera.

As already mentioned in Chapter 2, the depth map has pixels with values equal to zero corresponding to measurements not valid. To fill these "voids" in the depth map we decided to use the neighbour filter, which approximates a pixel with zero value with the closest pixel whose value is different from zero. Since filtering the entire depth map (which contains 512x424 pixels, with a lot of zeros) at each time step is very time consuming, our choice was to fill the zeros in the depth map with the neighbour filter only for those pixels which are required for the Occlusion Detection procedure.

Chapter 7

Experimental Validation

7.1 Introduction

In this Chapter we will present the experiments carried out to validate this work.

We are interested in assessing the performance of our work, in particular, focusing on the results achieved with respect to the following two aspects:

- The accuracy of the pose estimation of the ChArUco board using HoloLens. We are interested in characterizing how much the stability of the estimates is affected by the motion of this device when it is worn by the operator.
- The reliability of the estimate achieved through the application of the Particle Filter technique when the improvements proposed in Chapter 6 are applied. We are particularly interested in assessing the goodness of the estimated position and the volume in which particles are distributed.

7.2 ChArUco identification experimental validation

First of all we validated the accuracy of the estimate of the ChArUco marker pose using the standard OpenCV functions, which as outlined in [28], yields good results when the camera is kept fixed. We were interested in the performance using HoloLens since the movement of the user heavily affects the accuracy of the estimation, this would in turn prevent the correct alignment of the room model to the reference system of Kinect.

It is a known issue that the goodness of the recognition and pose estimation of a fiducial markers is strictly linked to the quality of the image taken. In particular, key aspects for an accurate estimation process are a good lighting, an high resolution image and a relative stillness of both camera and marker. To overcome problems arising by the lack of one or more of these conditions, several approaches are possible, like the one proposed in [6].

In our setup we were able to provide both a good lighting and high resolution images. Anyway, since HoloLens is a wearable device, the stillness of the camera cannot be guaranteed, considering that the user cannot be perfectly still due to physiological functions such as breathing. Therefore it is reasonable to assume that a slight blurring of the image will often be present.

Considering that HoloLens is not suitable for running computationally intensive programs to correct errors coming from the motion blur, we decided to use the standard assets and perform an evaluation of the stability of the measurements obtained from the device.

Therefore, to evaluate the loss of accuracy that wearing HoloLens would produce in the estimation process, we evaluated the performance obtained in three situations:

- keeping the device still on a support
- wearing it while standing
- wearing it while sitting

When HoloLens is worn the user tried to be as still as possible. We compared the results to understand how much the fact that HoloLens is worn affects the accuracy of the obtained measures.

The results of the tests are displayed in Figure 7.1. The accuracy of the detection was evaluated using as indicator the variance of the distance measurements along the Z axis of the camera, that is the direction along which the estimated positions are more spread. We analysed the accuracy of the estimated marker position as a function of the distance between the device and the actual marker during the detection.

Our focus was on the range in which the operator is more probably located during the detection, thus between one meter and one meter and sixty centimetres away from the board, but we considered also the

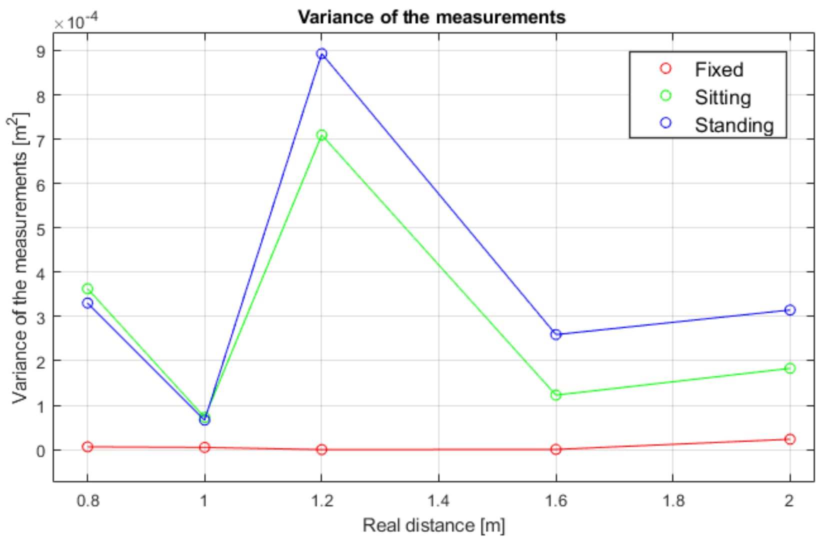

Figure 7.1: Variance of the ChArUco estimation using HoloLens

extreme case in which the span is two meters. More than 300 samples for each distance were taken in a similar lighting condition.

The results obtained, as illustrated in Figure 7.1, showed that the variance of the estimate when the device is fixed on a support is almost negligible. Differently, when HoloLens is worn the spreading of the data increases, but the result is considered acceptable.

A secondary but important effect highlighted by the results of the test is that an offset on the position measured along the Z axis of the camera is always present. This offset has a magnitude of 8 cm when the distance between user and camera is 0.80 m and decreases until it vanishes when the span is 1.60 m. We compensated this offset a posteriori, when composing the homogeneous transformation matrices.

We speculated that this offset is due to a light misalignment between the Z axis of the Hololens' main camera and the headset reference system.

Against our predictions, the difference between the variance of the samples acquired while sitting and while standing is not significant. Therefore the precautions taken to increase the accuracy were to perform the marker acquisition at a distance of 1.60 m from it and to try to be as still as possible, since placing HoloLens on a support, which would have been the best solution, in our case was not a viable option.

7.3 Particle Filter

We evaluated the performance of the Enhanced Constrained Particle Filter algorithm is a realistic assembly task where some occlusions, due to the motion of a robot and to the presence of certain geometries (i.e. a shelf), will occur from the perspective of the Kinect camera.

7.3.1 Setup

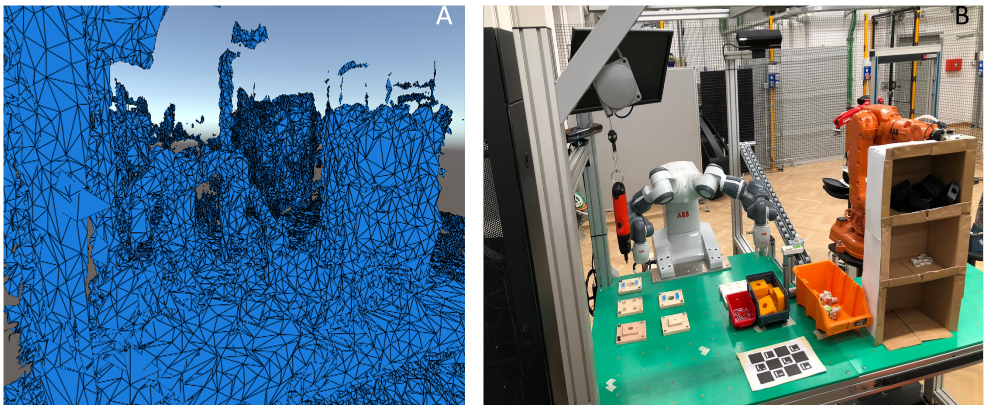

Figure 7.2: Workspace and its mesh representation

The setup for this experiment, shown in Figure 7.2.B from the user perspective, includes a cobot, several boxes and a shelf unit containing the components needed for the assembly procedure.

Figure 7.2.A displays the mesh reconstructed in Unity from the same perspective. Meshes were acquired using HoloLens according to the procedures explained through Chapters 4 and 5.

In Fig 7.3 the workspace is presented from the point of view of the Kinect and the sources of occlusion are highlighted. We assumed that in a realistic industrial scenario, two types of occlusions could occur:

- The one marked with number one 1 in Figure 7.3, which occurs when the human inserts his arm inside the shelf, is mapped a priori in the "Environment Reconstruction" phase and is hence present in the meshes describing the environment.
- The occlusion marked with number 2, which simulates an occlusion due to the motion of the robot arm. Thus, it cannot be neither

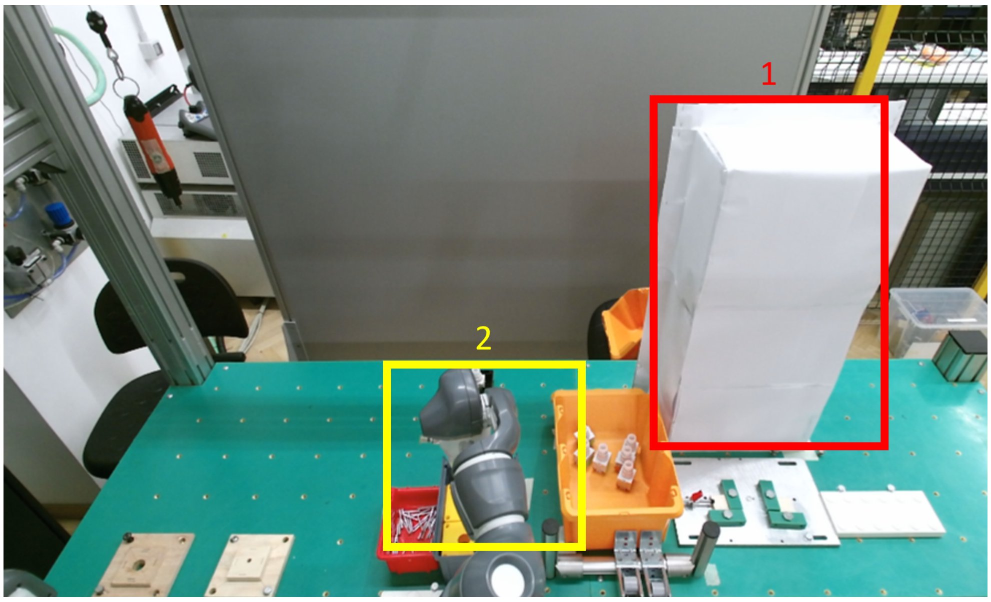

Figure 7.3: Workstation viewed from the Kinect

known a priori nor mapped through the HoloLens during the "Environment Reconstruction" phase.

We distinguished the occlusion typologies on the basis of their predictability and of their impact on the volume in which particles can propagate.

Considering the predictability two categories of occlusions can be distinguished:

- Static: an occlusion that is possible to map a priori of the working cycle and is fixed in the environment, hence its geometry is captured when the "Environment Reconstruction" is executed. Occlusions of this type can be caused by things like shelves, boxes or monitors, objects that are needed to perform the working activity and always present on the scene.

- Dynamic: an occlusion caused by something that was not possible to detect in the "Environment Reconstruction" phase. These can be generated by the robot itself interposing one of its links between the user and the camera, by a self-occlusion of the operator or by something being added in an unexpected way into the scenery.

On the other hand, also considering the impact of an occlusion on the volume of propagation of the particles two distinctions are possible:

- Complex: an occlusion in which the surrounding objects heavily affects the volume reachable by the particles preventing them from moving in at least two directions, thus heavily affecting the volume of propagation. An example is a situation in which a user moves its hand inside a concave object, like a shelf unit, in this case one of the outer surfaces of the object occludes the wrist while the others, like the upper and lower shelves, will prevent the particles propagation.

- Simple: an occlusion in which the surrounding geometry limits the particles in just one direction. This is the case when the wrist is occluded near a table. In this case particles are more free to evolve.

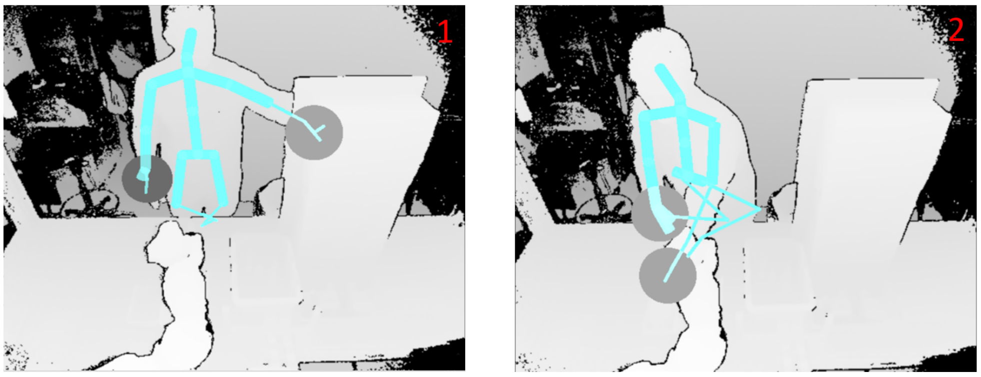

Figure 7.4: Occlusions highlighted in the Kinect's depth map

As shown in figure 7.4, in our environment the occlusion 1 is complex (since the operator has to introduce its hand between the shelves) and static, whereas occlusion 2 is simple and dynamic. We included both the typologies of occlusions multiple times in the same working activity to try to better highlight what difference the improvements implemented can make.

Regarding the algorithms tested we used three different variants of Particle Filter, progressively increasing the complexity:

- Original: the original Constrained Particle Filter algorithm described in Chapter 2, which include the Skeletal Distance and the Occlusion Detection verifications.

- Joint's Bounding Volume: this includes, beyond the capabilities of the Original algorithm, also the avatar generation and therefore the Joint's Bounding Volume described in Section 6.2.2.

- Complete: the Enhanced Constrained Particle Filter algorithm described in Chapter 6, that includes all the improvements described in this thesis. If a 3D model of the room and the associated homogeneous transformation matrices are not available, it falls back to the Joint's Bounding Volume variant, with exactly the same performances.

The activity which the operator is performing is the assembly of some components of an emergency button. During this procedure the tasks which lead to an occlusions are:

- The withdrawal of two components, a ferrule and half of the outer case of the button, from the shelf. Both these operations cause a static occlusion (from occlusion 1, as shown in Figure 7.4 A).
- The withdrawal of the other half of the outer case of the button and then of the screws needed to finish the assembly, these two subsequent operations cause two occurrences of dynamic occlusion (from occlusion 2, as shown in Figure 7.4 B).

7.3.2 Evaluation metric

As key performance indicators of our algorithm we used the following measures:

- The error between the estimate of the Particle Filter algorithm being tested and a reliable measurement of the wrist's position. To perform this test we removed the occlusion and used the Kinect to evaluate the position of the wrist to obtain a "ground truth". Then, during the experiments, the estimated position corresponding to the ground truth was stored. The distance from these two quantities was used as a measurement of the prediction error.

  The ground truth was chosen in a position corresponding to a complex static occlusion. This means that the Complete algorithm in its estimation procedure includes also the knowledge about the environment geometry. On the other hand, the Joint's Bounding Volume version will highlight what kind of performance can be reached without a model of the environment, just considering the modified model for the human body.
- The volume in which particles are spread, which is proportional to the uncertainty of the data. Since in literature, as in [12], the number

of particles which are close to the robot is used to determine the maximum speed at which it can work, we considered this measure directly related to the productivity.

Our choice of a significant volume was the confidence ellipsoid containing the 95% of the particles, an example of which is shown in Figure 7.5. The length of the semi-axes of this ellipsoid are proportional to the variance of the particles' distribution in the space and to the level of confidence that we want to obtain, thus capturing the dispersion of our samples.

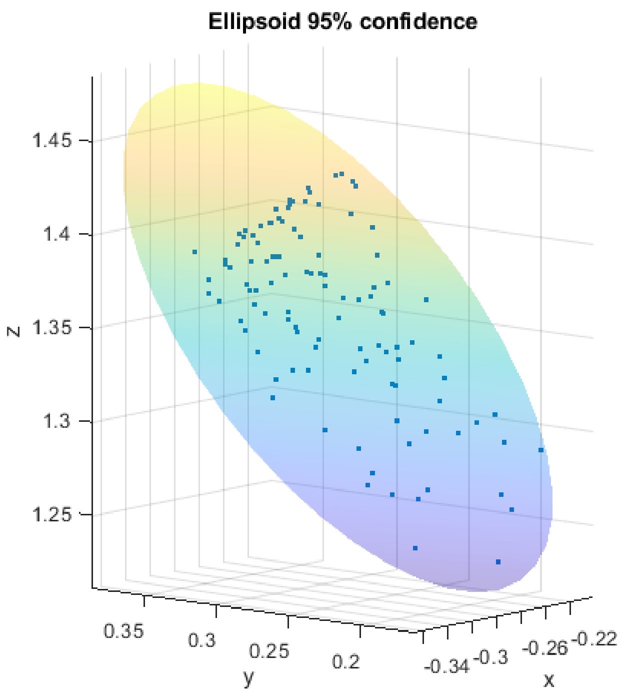

Figure 7.5: Example of confidence ellipsoid containing the 95% of the particles

It is important to notice that the procedure presented for the volume computation is carried out before the Resampling phase and doesn't include the particles which have a weight equal to zero, since they are in violation of the constraints.

7.3.3 Results

We collected data relative to 25 cycles per each algorithm considered (see Section 7.3.1). Each dataset contains all the data concerning one assembly procedure, during which two dynamic occlusions and two static ones occur.

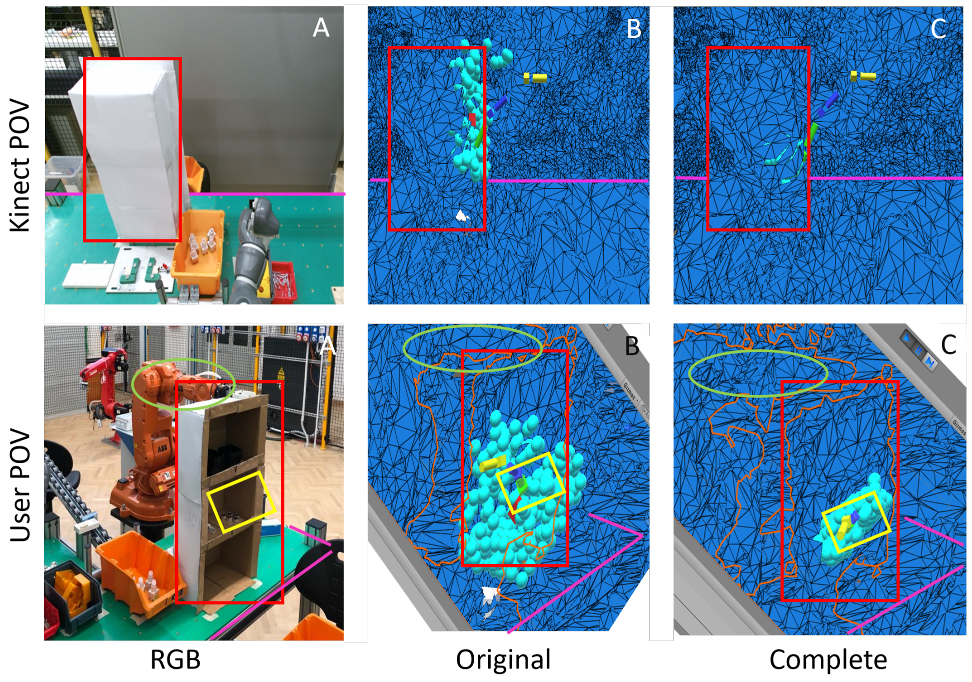

Figure 7.6: Graphical comparison between the algorithms when a static occlusion occurs

In Figure 7.6 is shown a qualitative comparison between the propagation region of the Original algorithm (denoted with B) and the Complete one (marked with C) under the condition of static occlusion viewed from two different perspectives. Since Unity meshes are difficult to interpret visually, the figures on the first column, marked with ”A”, present the best possible approximation of the same points of view. The upper layer of images shows what is seen by the Kinect, whereas in the lower one the user perspective is given.

In the images, the red rectangle is used to highlight the shelf unit, the pink lines to display the table contour, the green ellipses the position of the robot present on the background and, last but not least, the yellow square shows the goal region for the operator.

As can be seen the particles, rendered in Unity as light-blue spheres, are far more spread when the Original version of the algorithm is used, and they tend to propagate uniformly in the region not visible to the Kinect, this is generated both by the shelf and also in part by the operator.

On the other hand, the figures marked with C show that particles are well constrained inside the shelf unit (that was mapped a priori using HoloLens), coherently with the task that the operator was performing.

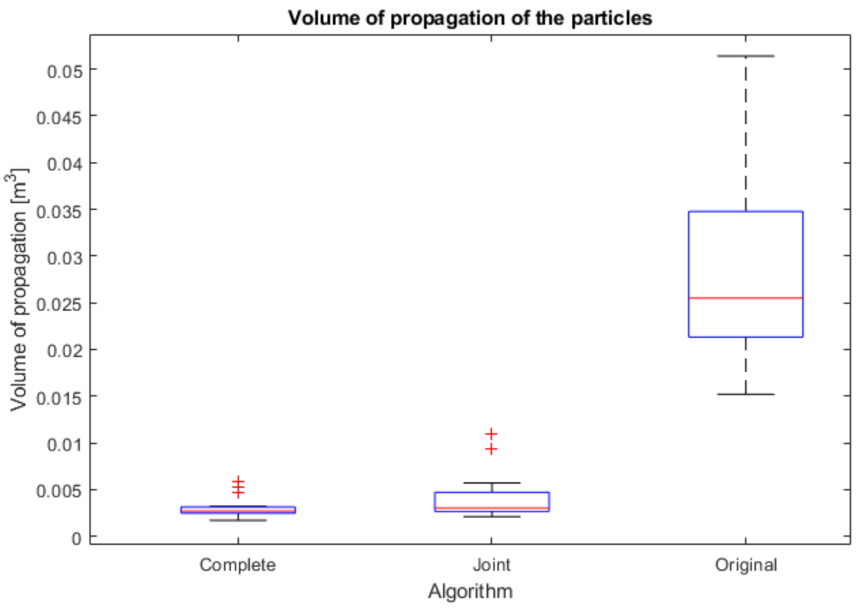

Figure 7.7: Boxplot representing the volume of propagation

The results obtained, displayed in fig 7.7, show a clear reduction of the volume occupied by the particles applying the Joint's Bounding Volume, which decreases of one order of magnitude the volume of propagation. In particular, also the variance of the result is diminished, because of the improved rejection of erroneous phenomena in the particle propagations, which will be detailed in Section 7.3.4.

Regarding the distance between real and estimated pose we found again that adding more constraints the estimation result is improved, as highlighted in Figure 7.8. Adopting the Joint's Bounding Volume algorithm the estimation error is reduced by the 21.7%, the Complete version of the Particle Filter further improves the result, reaching a reduction of the 28.3%.

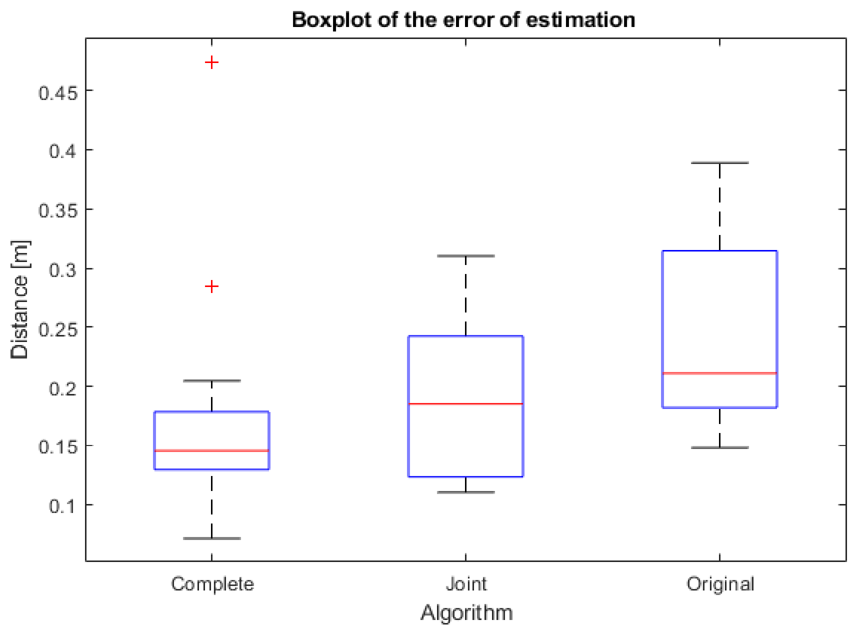

Figure 7.8: Boxplot representing the estimation error

7.3.4 Error rejection

Particularly interesting for us was the fact that the introduction of the virtualized human model prevented several errors present in the Original algorithm:

- In case of occlusion the particles tends to cumulate behind the body of the user. This happens because particles positions visible from the camera are eliminated. Being the surviving particles free to move on the spherical crown determined by the Skeletal Distance, they will tend to position behind the user, which with its body generated a non visible zone which respects the Skeletal Distance bound.
- If a hand is very close to the body, Kinect often misjudges its position and interprets this situation as an occlusion. In this case, exactly as in the previous, the particles will tend to cluster behind the user. Also the solution is the same.
- In case of occlusion particles tend to propagate uniformly behind the object hiding the joint. This tends to move the estimated position farther behind the occlusion unnecessarily.

All the mentioned errors are solved by the introduction of the Joint's Bounding Volume explained in Chapter 6, which effectively rejects this type of phenomenon.

Chapter 8

Conclusions

In this work we proposed a sensor fusion technique capable of relating the volumetric data, describing the operator's workspace, obtained using a Mixed Reality device, Microsoft HoloLens, to the information from a RGB-D camera, Microsoft Kinect, with embedded human tracking algorithms. We succeeded at this task by relying upon the simultaneous estimation of a fiducial marker pose using both the devices.

We then applied a customized constrained version of the traditional Particle Filter algorithm to address the problem of estimating the human wrist pose in the case it is unknown or uncertain, namely in the case of occlusions or noisy measurements.

To this aim we modelled the operator as a virtual avatar, whose movements were coupled to the actual human motions using the data acquired by Kinect. We constrained the motions of this human model by taking into account the joint's range of motion of the real arm. Moreover, we further added to the avatar the constraints provided by the physical environment where the human is located and works. To do that, we relied on the replica of the geometrical volumes of the workspace provided by the HoloLens in the form of meshes. We integrated the model of the environment, the avatar and the Particle Filter together inside a single application.

Thus, in this application the particles of the Particle Filter, representing the candidate positions for the operator's articulations, are constrained in their movements based on three elements:

- The shape and range of motion of the user's body, acquired via the Kinect and modelled in the avatar.
- The geometry of the workspace, captured in the room model coming

from the Mixed Reality device. Thanks to this model, it is possible to simulate collision between the particles and the physical surfaces.

- The regions visible to the Kinect. When an occlusion happens and the articulation being tracked is lost, the particles which represent positions of the joint visible to the Kinect are rejected. In fact, those particles are not coherent with the incoming data, which tell us that the joint is hidden.

We validated our method in a realistic industrial assembly task. The efficacy of our improvements was assessed in a test comparing our complete algorithm, with and without exploiting a model of the workspace, and the original formulation of the Constrained Particle Filter.

The results highlighted that our improved formulation of the Particle Filter is more accurate than the original version. In fact, with respect to the latter, it was able to:

- Reduce the estimation error by the 25% on average.
- Reduce the size of the volume of the distribution of the particles in the workspace by the 90% on average.

We considered the reduction of the particles dispersion particularly relevant, since in a collaborative scenario when the human pose can be estimated more precisely (with less uncertainty) the robot is less likely to be required to reduce its working speed unnecessarily. Thus, the significant reduction of the volume combined with the more precise estimation result is a key feature in increasing the productivity of the collaborative team.

Bibliography

[1] Radkowski R. ”Object Tracking With a Range Camera for Augmented Reality Assembly Assistance”. *Journal of Computing and Information Science in Engineering* 16.1 (2016): 011004.

[2] Muñoz Salinas R. ”ArUco: An efficient library for detection of planar markers and camera pose estimation”. (2019).

[3] Hubner P., Landgraf S., Weinmann M., Wursthorn S. ”Evaluation of the Microsoft HoloLens for the Mapping of Indoor Building Environments.” (2019).

[4] Mainetti R., Tironi A., Essenziale J. ”Microsoft Kinect v2 and Avateering”. *Course material.*

[5] Shao X., Biao H., Jong M. L. ”Constrained Bayesian state estimation–A comparative study and a new particle filter based approach.” *Journal of Process Control* 20.2 (2010): 143-157.

[6] Danying H., DeTone D., Malisiewicz T. ”Deep charuco: Dark charuco marker pose estimation.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* (2019).

[7] Gordon N. J.,Salmond D. J., Smith FM A. ”Novel approach to nonlinear/non-Gaussian Bayesian state estimation.” *IEE proceedings F (radar and signal processing).* Vol. 140. No. 2. IET Digital Library, (1993).

[8] Casalino A., Guzman S., Zanchettin A. M., Rocco, P. ”Human pose estimation in presence of occlusion using depth camera sensors, in human-robot coexistence scenarios.” *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, (2018).

[9] Chen N., Chang Y., Liu H., Huang L., Zhang H. "Human Pose Recognition Based on Skeleton Fusion from Multiple Kinects." *2018 37th Chinese Control Conference (CCC)*. IEEE, (2018).

[10] Ye M., Zhang Q., Wang L., Zhu J., Yang R., Gall J. "A survey on human motion analysis from depth data." *Time-of-flight and depth imaging. sensors, algorithms, and applications.* Springer, Berlin, Heidelberg, (2013). 149-187.

[11] Ragaglia M. "Towards a safe interaction between humans and industrial robots through perception algorithms and control strategies". Diss. Italy, (2016).

[12] Guzman S. "Human pose estimation in case of occlusions for safe and productive Human-Robot Interaction." Diss. Italy, (2017).

[13] Hübner P., Weinmann M., Wursthorn S. "Marker-based localization of the microsoft hololens in building models." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42.1 (2018).

[14] Kyriakides I., Morrell D., Papandreou-Suppappola A. "Multiple target tracking with constrained motion using particle filtering methods." *1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2005.* IEEE, (2005).

[15] Li H., Shrestha A., Fioranelli F., Le Kernec J., Heidari H., Pepa M., Spinsante S. "Multisensor data fusion for human activities classification and fall detection." *2017 IEEE SENSORS.* IEEE, (2017).

[16] Fatema Tuz Z., Md Wasiur R., Gavrilova M.. "Occlusion detection and localization from Kinect depth images." *2016 International Conference on Cyberworlds (CW)*. IEEE, (2016).

[17] Yoshida A., Kim H., Tan J. K., Ishikawa S. "Person tracking on Kinect images using particle filter." *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS).* IEEE, (2014).

[18] Xu Y., Xu K., Wan J., Xiong Z., Li Y. "Research on Particle Filter Tracking Method Based on Kalman Filter." *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, (2018).

[19] Frank B., Stachniss C., Grisetti G., Arras K., Burgard W. "Robotics 2 Camera Calibration." *línea]. Disponible en: https://goo. gl/HqTcj4.*

[20] Hentout A.,Aouache M., Maoudj A. "Key challenges and open issues of industrial collaborative robotics." *2018 The 27th IEEE International Symposium on Workshop on Human-Robot Interaction: from Service to Industry (HRI-SI2018) at Robot and Human Interactive Communication. Proceedings.* IEEE, (2018).

[21] Tuceryan M., Genc Y., Navab N. "Single-point active alignment method (spaam) for optical see-through hmd calibration for augmented reality." *Presence: Teleoperators & Virtual Environments* 11.3 (2002): 259-276.

[22] William Hoff. "Detecting Square Markers using OpenCV." *Course materials*, (2005).

[23] Thrun S. "Particle filters in robotics." *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., (2002).

[24] Mutto C. D., Zanuttigh P., Cortelazzo G. M. *Time-of-flight cameras and Microsoft Kinect (TM).* Springer Publishing Company, Incorporated, (2012).

[25] Ye M., Wang X., Yang R., Ren L., Pollefeys M. *Accurate 3D Pose Estimation From a Single Depth Image.*

[26] Ye M., Wang X., Yang R., Ren L., Pollefeys M. "Accurate 3d pose estimation from a single depth image." *2011 International Conference on Computer Vision.* IEEE, (2011).

[27] Coskun H., Achilles F., DiPietro R., Navab N., Tombari F. "Long short-term memory kalman filters: Recurrent neural estimators for pose regularization." *Proceedings of the IEEE International Conference on Computer Vision.* (2017).