Università degli Studi di Padova

Dipartimento di Ingegneria
dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea

Multi-frame techniques for long-term people re-identification with consumer depth cameras

Advisor: Prof. Emanuele Menegatti
Co-Advisor: Dott. Ing. Matteo Munaro

Student: Marco Carraro

13 Ottobre 2014
Anno Accademico 2013/2014

Abstract

In this work we address the people re-identification problem with particular attention to long-term re-identification in which a subject has to be re-identified even after some days from the last occurrence. In particular, we focus on multi-frame techniques, which exploit information from multiple frames for producing the re-identification output. We introduce three algorithms which improve classification performance obtained by state-of-the-art algorithms on two public datasets: IAS-LAB and BIWI RGBD-ID. These algorithms exploit 3D shape information obtained by consumer depth sensors like Microsoft Kinect. Since at every frame only the past history of a person is taken into account by these algorithms, they are all suited for online re-identification. Moreover, they are also efficient, in order to be executed in real time on-board of mobile robots.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

People re-identification is the problem of automatically recognizing and associating a subject to a track, after the person had been previously seen elsewhere. The solution of this problem can help different areas such as service robotics, health-care, video-surveillance and security application in general. Re-identification is divided into short-term re-identification and long-term re-identification based on the time passed after having viewed a subject. The first consists in recognizing a subject after few seconds/minutes having seen him, the latter consists in re-identifying a person after some hours or days having seen him, assuming that the person could have changed clothes in the meanwhile. The algorithms developed in this work fit in the latter category, thus cannot use some useful short-term information such as signatures composed of colors of some part of the body, the trajectory of the person and so on. As we describe in Chapter 2, the information useful for this particular task are very few and most of them are based on skeleton information returned by a depth sensor. In this work we will describe some algorithms which improve classification performance achieved by state-of-art algorithms by exploiting multi-frame techniques. This goal is reached using three different algorithms, the first one performs information fusion at the feature level while the others perform fusion at the decision-level. They are all based on the idea to make a decision not only relying on current data, but also looking at the decisions made in the past for the same track. In the remainder of this thesis, we will describe these algorithms and perform a comparison of our results with those of the best state-of-art algorithms.

1.1 Thesis structure

In Section 2, we will review the literature about people re-identification. We will cover both short-term and long-term re-identification.

In Section 3, we will give information about methods and libraries we have used.

In Section 4, we will describe a re-identification method based on the *Silhouette* of a person. This method does not perform very well compared with state-of-the art algorithms, but represents an efficient method which can be used in hard real-time applications.

In Section 5, we will describe the core part of our work, introducing three algorithms which exploit past history for re-identifying a person.

In Section 6, we will describe the datasets we have used for testing the performance of our algorithms and we will summarize our results comparing them to state-of-the-art results on the same datasets. Finally, we will report our conclusions and future works in Section 7.

Chapter 2

State of the Art

The people re-identification problem is a widely studied area. In literature, it is divided into two sub-problems: short term and long term re-identification. Our work addresses long term re-identification, but in the next sections we will give an overview of state of the art for both categories.

2.1 Short-term re-identification

Short term re-identification is defined as the capability to re-identify a person's track few seconds or minutes after having lost it. In literature, a wide set of algorithms exist. Here below, we subdivide these techniques according to the type of feature they use for re-identification.

1. **Algorithms based on color information**: Colors can give an extremely important information of a track. In fact, difference of clothes and skin can often be sufficient to classify one or more tracks. The main problem of this technique is the choice of the keypoints where to compute colors descriptors. These keypoints are highly informative points extracted with some techniques. Descriptors of these points are often computed with 2D algorithms like SURF[4], SIFT[13], BRIEF[6], ORB[19] or similar ones. There are similar algorithms also for 3D data, such as PFH[22], PFHRGB which add color information to the previous algorithm, FPFH[21], SHOT[24], and SHOTRGB[25].

2. **Algorithms based on trajectory information**: Also trajectory can give some information. If a track is lost and after few milliseconds a new track has been detected in a position that is congruent with the trajectory the previous track had, they're probably the same track. As

reported in [8], trajectory information is used mainly for re-identifying a lost track after few seconds in a people tracking system.

3. **Algorithms based on person's biometrics features**: There a wide category of biometrics signatures that can correctly re-identify a track. Some examples of this biometrics signatures are: face recognition[26], iris[27], fingerprints[10] and so on.

There are a lot of works that combine one or more of this techniques to perform re-identification. Here we cite the most famous and recent ones.

In [16], authors calculate color descriptors only on human joints composing a signature with them. The position of every joint is returned by the skeletal tracker which can compute them efficiently. This results in a robust algorithm for short-term re-identification. The problem with this method is related to the skeletal tracker which often cannot track every joint.

In [7], authors try to divide each subject in parts, calculate local descriptors for each part and then combine them to form a signature. Every local descriptors calculate the overall chromatic content, the spatial arrangement of colors into stable regions, and the presence of recurrent local motifs with high entropy. This work is robust even in presence of low resolution images and with illumination changes.

In [17], short-term re-identification is necessary for the tracking purpose of re-assigning a track after having lost it. In this work the authors use an online version of the Adaboost classifier([9, 14]), where the descriptors was composed of two parts picked in the color histograms of every track. This classifier is particularly performant because histograms are calculated only one time per frame for all the features used.

2.2 Long-term re-identification

Long term re-identification is defined as the capability to correctly re-identify a person days after having seen it. This field is more difficult than the previous one, because information such as color cannot be used due to clothes changing, trajectory lose of importance and biometrics are often difficult to use in a non-collaborative environment or if a subject is far from the sensor due to resolution problems.

Instead of short-term re-identification that has been widely studied, long-term re-id is currently an open-world for researchers. We describe here some works which try to give a solution to this problem.

In [1], the authors build a tridimensional model of every subject similar to a sarcophagus. Later these models can be compared and verified. The model is built taking hundreds of vertices and linking them making a monolithic hull. The re-identification goal is then computed using a weighted sum of Hellinger distances between vertices of training and testing models. This method works also with few and low resolution images.

In [2], the authors propose some *soft-biometrics* features that are defined as distances between particular points in a person point cloud. The features considered are the following:

- d_1: Distance between floor and head
- d_2: Ratio between torso and legs
- d_3: Height estimate
- d_4: Euclidean distance between floor and neck
- d_5: Euclidean distance between neck and left shoulder
- d_6: Euclidean distance between neck and right shoulder
- d_7: Euclidean distance between torso center and right shoulder
- d_8: Geodesic[1] distance between torso center and left shoulder
- d_9: Geodesic distance between torso center and left hip
- d_{10}: Geodesic distance between torso center and right hip

After computing this features with RGB-D sensors like Microsoft Kinect and NiTE Skeletal tracker, they are combined in a signature. Then, in the re-identification phase, a weighted sum of common features between two different signatures is computed obtaining a distance measurement between signatures.

In [15], a tridimensional model of a freely moving person with Microsoft Kinect is built. The model is created matching different point clouds of

[1] A geodesic distance is a distance between two points following the surface of the model

the same person after having warped them in a standard pose. After this training phase the re-identification task is made by means of the Iterative-Closest-Point algorithm (see 2.2.2.1) which returns a fitness score which is a similarity distance between the two clouds. The algorithm presented in this work has been named Point Cloud Matching (PCM). Our work is based on this article so, in the next section, we will explain better every phase of this algorithm.

2.2.1 Re-Identification by means of point cloud shape

2.2.1.1 Model creation

Algorithm 1: Model Creation Algorithm

Data:

- Set of k standard poses (position and orientation)
- A multi set containing n set of frames everyone related to one subject free to move filmed by Kinect sensor.

Result: A set of n models $M = \{M_1, M_2, ..., M_n\}$

```
for current_model ← 1 to n do
    M_current_model ← ∅
    for * current_frame in frames of current_model * do
        P_current_frame ← * point_cloud of current frame *
        * Divide the points of P_current_frame into body parts *
        for b in * body parts of P_current_frame* do
            * transform body part in standard pose *
        end
        P_current_frame ←computeMLS(P_current_frame)
        P_current_frame ←computeVoxeling(P_current_frame)
        P_current_frame ←performICP(P_current_frame,M_current_model)
        * add P_current_frame points to M_current_model*
    end
end
```

The first phase is a training phase in which person's models are built. Every subject is free to move in a scene in front of a Microsoft Kinect sensor. For every frame the person's point cloud is extracted, together with position and orientation of the skeleton joints. The joints returned by the Kinect can have three confidence states, *not_tracked*, *inferred* or *tracked*. If at least one of the joints is *not_tracked* or *inferred*, the frame is discarded. On the other hand, if the frame has all the joints tracked, the acquired point cloud has to be warped to standard pose. This can be done by a segmentation of body parts; a point p belongs to a body part q if $q = argmin_{l \in L} \|p - l\|$ where L is the set of person's links. Every body part has to be transformed in standard pose, which is simply a similar pose in which every cloud is directly comparable. To perform this transformation, every part is rototranslated according to its joint current pose and the standard orientation of the same

joint. If Q_c is the orientation of a body part in the current frame and Q_s is the same joint in the standard pose of the same person than the whole rotation to apply to the points of the part is $R = Q_s(Q_c)^{-1}$. After this transformation we have a point cloud directly comparable with the others, so the next steps the authors do is to refine the frame's standard point cloud with the Moving Least Square algorithm (see 2.2.1.2), a downsampling (Voxeling) of the cloud for timing problems and the match with the Iterative Closest Point of the cloud to the current model, adding the points to it. In this way it is possible to make a 3D model of every subject while freely moving in the scene. The whole algorithm pseudocode is reported in Algorithm 1.

2.2.1.2 Moving Least Square

The Moving Least Square algorithm [12] is an approximation algorithm that tries to reconstruct a continuous function from a discrete quantity of points by upsampling or downsampling. It is very used in computer vision in particular for reconstructing surfaces from point cloud. The Moving Least Square algorithm used is the one implemented in the *Point Cloud Library* inside the "surface" module. This method is used in the Point Cloud Matching Algorithm for smoothing purpose.

2.2.1.3 Voxeling

In the PCM algorithm, when a point cloud of a frame is acquired, after the smoothing phase, it is necessary to downsample its points. The ICP algorithm (see 2.2.2.1), in fact, requires to rototranslate all the points of the cloud, at every iteration. These steps could make ICP extremely slow if the input cloud has too much points. A cloud downsampling with the Point Cloud Library can be done by Voxeling. This method divides the point cloud into small 3D boxes (voxels) with a certain edge size (called voxel size). Then, for every voxel, it computes the mean point of the cloud points in the voxel. These mean points compose the final point cloud.

2.2.2 Point Cloud Matching (PCM) algorithm

2.2.2.1 Iterative Closest Point

The re-identification task of [15] is done with the Iterative Closest Point (ICP)[5] algorithm which compares each test frame with every model saved in the training phase. This algorithm is used in computer vision to register two similar point clouds. Its goal is to minimize the distance between the

Algorithm 2: Point Cloud Matching Algorithm

Data:

- Set of M models of n subjects freely to move computed by algorithm 1
- A multi set containing n set of test frames everyone related to a subject filmed by Kinect sensor.

Result: A set of n classification matrices $C = \{C_1, C_2, ..., C_n\}$ everyone with N_k rows and n columns where k is the index of the model considered and N_k is the number of test frames representing subject k

```
for * current_frame in frames of current_model * do
    P_current_frame ← * point_cloud of current frame *
    * Divide the points of P_current_frame in body parts*
    for b in * body parts of P_current_frame* do
        * transform b points in standard pose*
    end
    P_current_frame ←computeMLS(P_current_frame)
    P_current_frame ←computeVoxeling(P_current_frame)
    for m ← 1 to n do
        fitness_scores ← computeReIdentificationScores(P_current_frame, M_m)
        * fill row current_frame with fitness_scores in the classification matrix C_m*
    end
end
```

two clouds via a mean-square distance metric. This can be done following these passages:

- **Compute closest points of the clouds**: closest points are points which have the smallest euclidean distance from other cloud points.
- **Compute the registration**: compute the roto-translation to the closest point to reduce the euclidean distance
- **Apply the registration**: apply the registration to all the points of the cloud
- **Repeat** the above points until the mean square distance is above a threshold or until a maximum number of iterations is reached.

It is demonstrated that the algorithm converges to the minimum distance between the clouds, but the number of iterations needed can be very large. However, in [5], authors show that the first iterations compute a large roto-traslation of the first cloud, then the remaining iterations refine with small changes the registration. So, if the input clouds are close to each other, it can be often sufficient to give a not so high maximum number of iterations to the algorithm, for obtaining a very good registration.

2.2.2.2 The algorithm

The Point Cloud Matching algorithm requires models created by algorithm 1 and returns a classification matrix C for each model given. This algorithm is pretty similar to algorithm 1 in which models are created. Every frame is transformed in standard pose and, after a MLS and a Voxeling, it is matched with every model. The method *computeReIdentificationScores* computes the *fitness scores* between the frame cloud and the models using ICP. These scores represent the similarity between the frame cloud with every model in the gallery. These values are put in a row in the final matrix. The whole algorithm pseudocode is reported in algorithm 2

Chapter 3

Materials and Methods

In this section we will describe materials, libraries and conventions adopted in this thesis. The algorithms we have implemented are written in `C++` and tested with Ubuntu 12.04.

3.1 Robot Operating System (ROS)

ROS(Robot Operating System)[1][18], is the middleware we have used in our work. It has been released to provide a unique framework for robotics. It comes with a wide collection of state-of-the-art algorithms ready to use and hides sensor configuration to the end user. Furthermore it has a modular organization and provides a standard way to send messages.

ROS architecture can be visualized as a peer-to-peer graph in which ROS nodes (processes) communicate between each other with messages. These are the main concepts in ROS:

- **Node**: A node in ROS is a running process. It can do operations, algorithms and send/receive messages from/to other nodes.
- **Message**: A message is a data structure representing data that nodes send/receive to/from each other. There are primitive messages like integers, strings, vectors etc. and it is possible to define own messages, composed of one or more primitive messages.
- **Packet**: ROS is organized into packets; a packet is the set of nodes and code relative to a task. A user can develop his own packets that

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

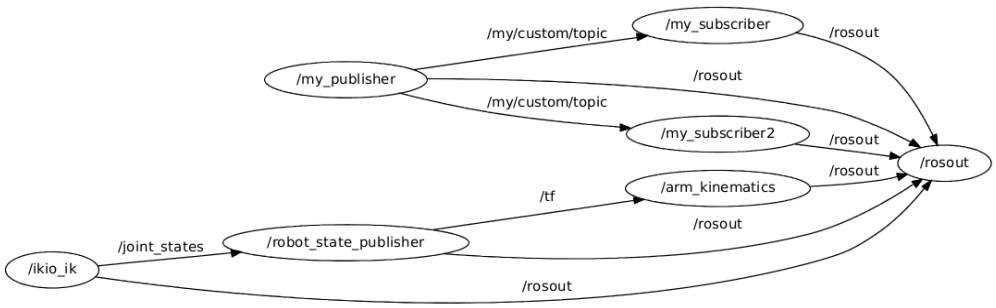


Figure 3.1: An example of ROS graph

can be integrated with other packets freely downloadable from ROS community websites.

- **Topic**: A topic is a named bus which is used by nodes to send/receive messages by publish/subscribe primitives. Every topic is associated to a message file and can be used by more nodes at the same time.

- **Service**: A service is an alternative way of communication in ROS that does not use topics. It's thought for request/reply RPC interactions. Services are composed of two messages one for the request and the other for the reply, and it is defined by a name through which is called by nodes.

- **Bag**: ROS gives the possibility to record one or more topics. These recordings are called *bags*, and can be played in a second time. When a bag is played the recorded topics appear in the ROS graph, making possible the interaction with them.

In figure 3.1 we report an example of ROS graph in which nodes are mapped into nodes of the graph and topics are mapped into edges. It is a directed graph because a node can subscribe a topic (ingoing arc) and publish a topic(outgoing arc).

3.2 Point Cloud Library

The Point Cloud Library(PCL)[2][20] is an open source library that provides many algorithms for processing N-th dimensional image data. It is very used in computer vision and in robotics for robot perception, it is Open-Source and it is well integrated in ROS. The library contains state of the art algorithms for: filtering, feature estimation, surface reconstruction, registration, model fitting, segmentation and many others.

In this work, every point cloud we take from the Kinect sensor is manipulated with this library. We used it for these things:

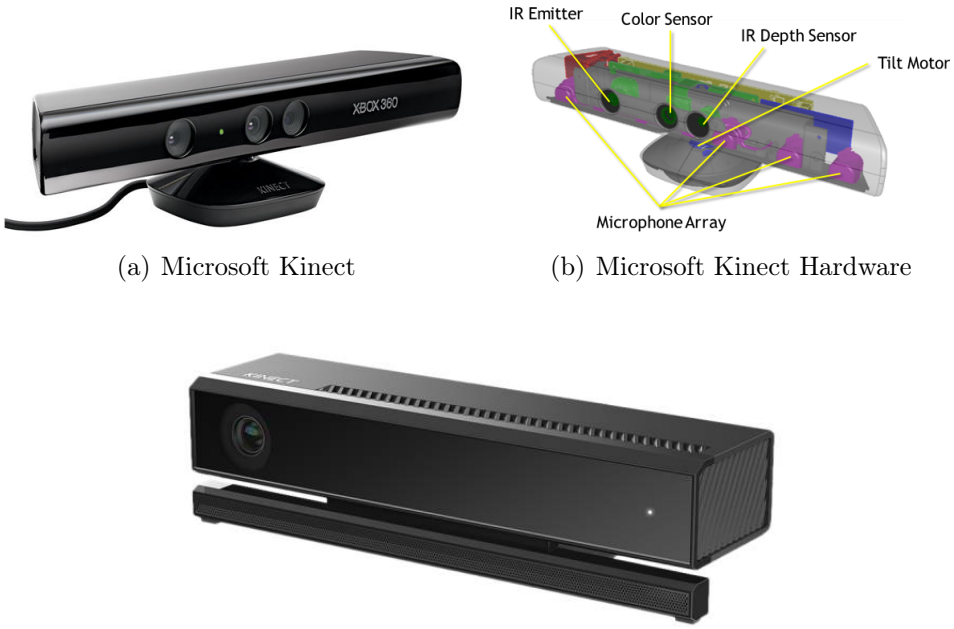
- Acquiring and saving point clouds
- Down-Sampling (Voxeling) a point cloud
- Calculating distance between points in a cloud
- Performing the ICP algorithm (see section 2.2.2.1)
- Visualizing clouds

3.3 Microsoft Kinect

Microsoft Kinect (fig. 3.2(a) and 3.2(b)) is the depth camera we have used for this work. It is a camera that provides depth combined with RGB classic information. Depth information comes from a range sensor developed by *PrimeSense*, an Israelian company. This sensor is based on an infrared pattern projection technology which measures depth based on how the pattern is deformed in the scene. It allows computers to directly view in three dimensions, and, for this reason, it has been largely used by universities for research purpose. Acquisition of the depth data suffers in presence of infrared light that can disturb the view of the projected pattern. For this problem, Microsoft Kinect sensor cannot be used outside or in presence of infrared light sources. Also black objects cause some problems with the depth data due to absorption of the light.

Such explained in [11], Microsoft Kinect can extract depth information by a triangulation process. The laser projects a beam of infrared light which is split into multiple beams by diffraction creating patterns of speckles. These

[2]http://www.pointclouds.org



(a) Microsoft Kinect

(b) Microsoft Kinect Hardware

(c) Microsoft Kinect v2

Figure 3.2: Microsoft Kinect sensors

speckles will appear shifted in the scene filmed by Kinect, basing on the position of the objects. Kinect sensor obtains a disparity map comparing a reference pattern with the one obtained by the current scene. In Figure 3.3 is explained the relation within the distance of a point k and the measured disparity d. The coordinate system of the figure has Z perpendicular to the image plane, X axis orthogonal to Z along the baseline b and Y orthogonal to Z and X like a right handed coordinate system. Assume that an object in the reference plane was at Z_0 distance with the sensor and that the disparity measured was d, then by the triangles similarity:

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \tag{3.1}$$

and:

$$\frac{d}{f} = \frac{D}{Z_k} \tag{3.2}$$

Where Z_k is the depth distance of the point k we want to measure, D is the displacement of the point k, b and f are respectively the base length and

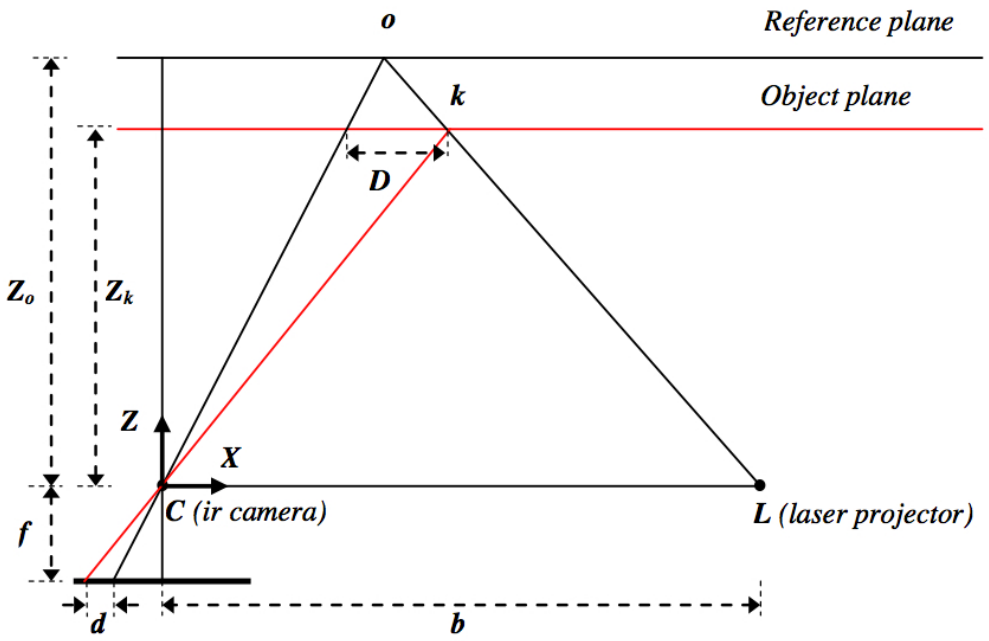


Figure 3.3: Depth and disparity relations

the focal length of the infrared camera and d is the observed disparity. From equations 3.1 and 3.2 with trivial passages we can finally obtain Z_k:

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{fb}d} \tag{3.3}$$

In late 2013, Microsoft released Microsoft Kinect v2 (fig. 3.2(c)). This sensor adopts a new technology for retrieving depth information based on time-of-flight technology. The new sensor estimates a depth matrix measuring the time of flight of an emitted signal. This technology can be reliable also in outdoor scenes, but seems to have problem with black objects like the previous one and introduce noisy pixels in presence of particular reflecting objects like coins or mirrors.

3.4 Skeletal Tracker

Skeletal trackers allow us to extract information about skeletal joint positions and orientations of one person in a scene. These information are very

important for some algorithms described in Section 2, in fact, these points can be used to build a person signature.

There are several skeletal trackers available, we describe here the most famous and used:

- **Microsoft Skeletal Tracker** [23]: With Kinect sensor, Microsoft also developed a skeletal tracker, included in the SDK freely downloadable from the Microsoft website[3]. This tracker can track 20 frontal person's joints (see fig. 3.4(a)) at 30 fps. It can also track until 2 people joints together in the scene and recognizes until six people together. The working distance of the tracker is from 0.5m to 3.5m and it can provide also the orientation of the joints (absolute or relative).

  With the Kinect v2, a new skeletal tracker has been released. This sensor can track 25 joints (fig. 3.4(b)) introducing new joints for hands and neck.

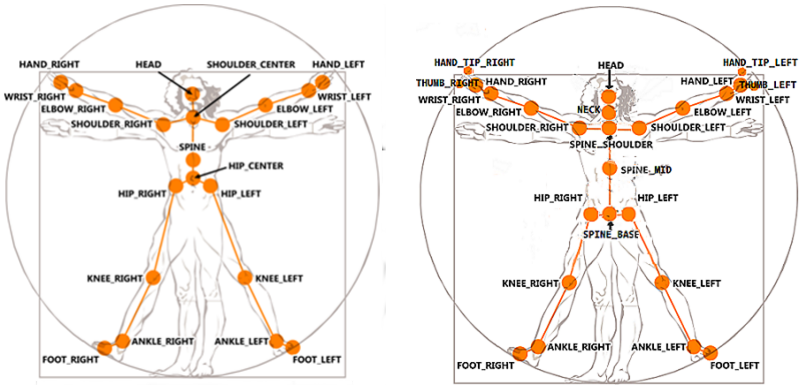
- **NiTE Skeletal Tracker**: This skeletal tracker was developed by OpenNI (Open Natural Interaction). It can track 15 joints of people in a scene. It is less precise than Kinect Skeletal tracker, but it can track joints also when a person is backward and it can be included from ROS.

- **PCL Skeletal Tracker**: Also in the Point Cloud Library [20], a skeletal tracker exists. However, this tracker requires a Nvidia Graphic Card compatible with Nvidia CUDA and it is less precise than the previous ones. For these reasons, this tracker has not been used in our work.

3.5 Face Detection

The Point Cloud Matching algorithm uses a face detector algorithm [26] for estimating when a subject is frontal or not. In the classic algorithm this information is used only for a validation check on the skeleton of the subject, deciding if the frame is good for classification or not. In our algorithms we used the face detector also to differentiate the classification sums and the multi-frame models (see 5.1 and 6.4). In this way, we have noticed that we can improve classification performance of the PCM algorithm.

[3] http://www.microsoft.com/en-us/kinectforwindows/



(a) Kinect Skeletal joints

(b) Kinect v2 Skeletal Joints

(c) NiTE Skeletal Joints

Figure 3.4: Skeletal Trackers

Chapter 4

Silhouettes based Techniques

In this work we have introduced the concept of *Silhouette*. A silhouette can be viewed as a sub-sampling of a 3D model that describes a side view of a person. This is represented by a very small subset of the original model's points so computations with them are extremely efficient. In this section, we will formalize what silhouettes are, how we construct and use them to perform long term re-identification.

4.1 Silhouette

A silhouette represents the side view of a person. Before introducing it we have to give some definitions.

Def. : Given a model M, $x_0, y_0 \in \mathbb{R}, \epsilon, \gamma \in \mathbb{R}^+$ we call $bucket(x_0, \epsilon, y_0, \gamma, M)$ a set of points: $B = \{P = (x_P, y_P, z_P) \in \mathbb{R}^3 \mid x_P \in (x_0, x_0 + \epsilon) \wedge y_P \in (y_0, y_0 + \gamma) \wedge P \in M\}$.

A bucket represents a set of points of a model with limited x and y components. In every bucket there are points with similar x and y components. Now, we have to extract a subset of these points which better represents the front side view of a person than the whole bucket's points do.

Def. Given $\lambda \in (0, 1)$ we call *important points* of a bucket B a set of points $I_p(B, \lambda) = \{P = (x, y, z) \in \mathbb{R}^3 \mid z \in (Z(1 - \lambda), Z)\}$ where Z is defined as: $Z = \{z \mid p = (x, y, z) \in B \mid \forall \widehat{p} = (\widehat{x}, \widehat{y}, \widehat{z}) \in B \Rightarrow z \geq \widehat{z}\}$

The important points have the greatest z components in the bucket. Every important points set $I_p(B, \lambda)$ is then represented by a single mean point $p_\mu(B) = (x_\mu, y_\mu, z_\mu)$ with these components:

$$x_\mu = \frac{\epsilon}{2}$$

$$y_\mu = \frac{\gamma}{2}$$

$$z_\mu = \frac{\sum_{p=(x,y,z)\in I_p(B,\lambda)} z}{|I_p(B,\lambda)|}$$

At this point we have defined buckets and mean points that represent them. A silhouette is a particular set of these mean points with buckets picked on the front of the model. We now define what we mean with front of the model, using minimal bounding box of a model M.

Def. : Given a model M, a bounding box of M, written P_M, is a rect parallelepipedon with this property:

$$P_M(B) = \{\forall q \in M \Rightarrow q \in P_M(B)\} \tag{4.1}$$

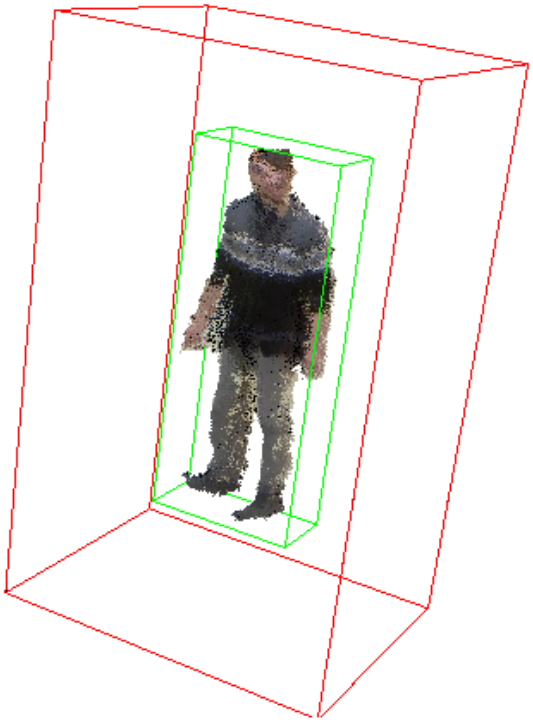


Figure 4.1: A model cloud with two bounding boxes, the green one is the minimal

We are interested in the *minimal bounding box* of the model M, which is simply a bounding box that, for every face, it has at least one point of the

model M that stays on the face.

In figure 4.1 a point cloud of a person (the model) with two bounding boxes around it can be viewed. The green one is the minimal bounding box. Now we are interested in selecting a family of buckets $B = \{B_1, B_2, ..., B_n\}$ along the middle column of the front face of the bounding box.

Now we are ready for the silhouette definition:

Def. : Given a family of buckets of a model M

$$B_M = \{B_1(x_1, \epsilon_1, y_1, \gamma_1), B_2(x_2, \epsilon_2, y_2, \gamma_2), ..., B_n(x_n, \epsilon_n, y_n, \gamma_n)\}$$

along the middle column of the frontal face of the model's minimal bounding box P_M of the model M, with the following property:

$$y_i + \gamma_i = y_{i+1}$$

$$\forall 1 \leq i < n, x_i = x \wedge \epsilon_i = \epsilon$$

We call *Silhouette* of the model M and we write S_M, the set of n mean points of the family of buckets $B = B_1, B_2, ..., B_n$.

In figure 4.2 a set of figures are shown, that represent silhouettes of a model with different parameters.

4.2 Silhouettes Matching

Silhouettes are light-weight point clouds that represent the side view of a person. The algorithm we have developed is described in Algorithm 3.

Algorithm 3: Silhouette mean distance

Data:

- Set of gallery models $M = \{M_1, M_2, M_3, ..., M_n\}$ of n people
- Set of test frames $F = \{F_1, F_2, ..., F_m\}$ representing the same n people for whom we have a model, but in different days

Result: A rxn (r is the number of skeleton valid frames on m frames) classification matrix C in which at i-th row ($1 \leq i \leq r$) we have n value, the j-th ($1 \leq j \leq n$)represent the mean distance between silhouette of the frame i and the model j

```
M initialization and computing of S_M the family of models' silhouettes
j ← 1
for i ← 1 to m do
    F_i ← current_frame
    Sk_i ← getSkeleton(F_i)
    if Sk_i is_valid then
        j ← j + 1
        T_i ← getPointCloud(F_i)
        ST_i ← computeStandardPose(T_i)
        S_i ← computeSilhouette(ST_i)
        for k ← 1 to n do
            * S_i = {P_1, P_2, ..., P_t} points of the current silhouette *
            sum ← 0
            for l ← 1 to t do
                * S_Mk = {PM_k1, PM_k2, ..., PM_kt} points of the current silhouette *
                sum ← sum + z (PM_l) - z(P_l)
            end
            C_jk ← sum /t
        end
    else
        continue
    end
end
```

This algorithm is pretty similar to the one used in [15]. The only difference is the use of the silhouette instead of the full person point cloud. The output of the algorithm is a rxn classification matrix, where r is the number of valid frames from the total m frames and n is the number of models in the gallery. The (i, j)-th element of this matrix contains the mean distance of the i-th frame's silhouette to the j-th model's silhouette. This algorithm is the first we have tried in our work, but we have obtained bad results wrt reference article [15]. The main problem is the matching of the silhouettes between test frames and gallery models. The test frames silhouettes are not so close to gallery models silhouettes due to noise in frame acquiring. With new sensors like the Microsoft Kinect v2 combined with Multi-frame PCM algorithm the test silhouettes should be more similar to gallery ones and the algorithm should perform better.

(a) A model obtained with Algorithm 1

(b) Silhouette of the model with 300 points

(c) Silhouette of the model with 150 points

(d) Silhouette of the model with 75 points

(e) Silhouette of the model with 50 points

Figure 4.2: Model and silhouettes

Chapter 5

Multi-frame based Techniques

In this section we will explain the novel techniques we have introduced in this work. Current re-identification methods make the classification of a frame only upon its characteristics. In our algorithms the decision on a frame is based on its characteristics and also on the previous decisions made for the track that has generated that frame. In fact the tracking information is not difficult to obtain, and can make the classification more stable specifically in presence of noisy frames.

We propose three different algorithms representing different voting scheme. They are all based on the PCM algorithm already described (see 2.2.2). This is a list of the algorithms:

- **Multi-frame PCM**: This algorithm is based on the idea to match not only a single frame against the gallery models, but a test model, built with the last $K-1$ frames seen and the current one. This algorithm represents a classifier based on information fusion at the feature level.

- **LastK PCM**: This algorithm, at frame i, takes as i-th row of the classification matrix, the sum of the last $K-1$ rows of the classic PCM classification matrix plus the classification scores of the i-th frame with the PCM algorithm. This algorithm is based on information fusion at the decision level.

- **RankingBased PCM**: This algorithm is similar to the previous one, but instead of scores it takes into account only the rank gained by every model. At frame i, it makes a sum of the classification ranks of the last $K-1$ frames and the current one taken in the PCM classification matrix and it writes the results on the i-th row of the classification

matrix. This algorithm is based on information fusion at the decision level.

5.1 Multi-frame Point Cloud Matching

The Point Cloud Matching algorithm (sez. 2.2.2), matches every single test frame with the models in the gallery set, which are built with a wide collection of training frames. The main idea of the Multi-frame PCM algorithm, is to build a *test model* by registering together the test frames previously seen. With this approach, we obtain a *test model* which can be compared with gallery's models by means of the classic PCM algorithm.

5.1.1 ModelCreation

How we create models and manage frames inside of it is a critical aspect for obtaining good results. In particular, we have to define how many frames to keep in the test model, how to align them and how to remove a frame if the maximum number of frames in the model is reached and a new one has to be inserted.

A model is a set of previously registered frames, so when we receive a new test-frame from the sensor we do the classic operations (*computeStandardPose*, *MovingLeastSquare* and *StatisticalOutlierRemoval*), and we do an Iterative Closest Point on the test-model adding the registered frame points to the model, taking track of the fitness score returned by ICP. The first frame of a model is added without the ICP step, and the fitness score saved is the best from the matching with gallery's models.

When the limit of frames in the test model is reached, the points of the worst frame are removed (based on fitness scores) and the next frame is added to the test model. If a frame has a too low fitness score with the test model, it is discarded.

5.1.2 Multiframe PCM Algorithm

The algorithm pseudocode is listed in Algorithm 4. The only difference with the classic PCM Algorithm (see algorithm 2), is the presence of the test model and the addFrame method.

Algorithm 4: Multi-frame Point Cloud Matching Algorithm

Data:

- Set of M models of n subjects freely to move
- A multi set containing n set of test frames everyone related to a subject free to move filmed by Kinect sensor.

Result: A set of n classification matrix $C = \{C_1, C_2, ..., C_n\}$ everyone with N_k rows and n columns where k is the index of the model considered and N_k is the number of test frames representing subject k

```
test_model ← ∅
for * current_frame in frames of current_model * do
    P_current_frame ← * point_cloud of current frame *
    * Divide the points of P_current_frame in body parts*
    for b in * body parts of P_current_frame* do
        * transform b points in standard pose*
    end
    P_current_frame ←computeMLS(P_current_frame)
    P_current_frame ←computeVoxeling(P_current_frame)
    score ← computeICP(P_current_frame, test_model)
    test_model ← addFrame(P_current_frame, score)
    for m ← 1 to n do
        fitness_scores ← computeReIdentificationScores(test_model, M_m)
        * fill row current_frame with fitness_scores in the classification matrix C_m*
    end
end
```

As explained before, *addFrame* makes possible to add a frame to the test-model following the rules in section 5.1.1. It simply adds the current frames points to the test-model if the maximum number of frames has not be reached, otherwise, it removes the points of the worst frame before adding the new one.

This algorithm is based on the idea to match a training model with a testing model, obtained by combining past information of the person and by making a so-called information fusion at the feature-level. With this algorithm, we have added two parameters to the classical one: the maximum number of frames in the test model and the maximum value of score allowed to a frame to be inserted in the test model.

5.2 Last K Point Cloud Matching

This algorithm (alg. 5) implements a classifier based on information fusion at the decision-level. The idea behind it is to give a weight to every decision made by the PCM algorithm at every frame. The weight that is natural to use is the fitness score of every frame with every model in the gallery obtained with the standard PCM algorithm. Thus, frame i will be matched

with gallery models obtaining a row of fitness scores for every model. This row is not filled directly in the matrix, but is previously summed with the last $K-1$ rows of the standard matrix. This point is important because if we sum the last $K-1$ rows of the current matrix instead of the standard one, we make a different operation, giving more importance to each frame. This algorithm has only one parameter to set (K).

Algorithm 5: LastK Point Cloud Matching algorithm

Data:

- Set of M models of n subjects freely to move
- A multi set containing n set of test frames everyone related to a subject filmed by Kinect sensor.

Result: A set of n classification matrices $C = \{C_1, C_2, ..., C_n\}$ everyone with N_k rows and n columns where k is the index of the model considered and N_k is the number of test frames representing subject k

```
for * current_frame in frames of current_model * do
    P_current_frame ← * point_cloud of current frame *
    * Divide the points of P_current_frame in body parts*
    for b in * body parts of P_current_frame* do
        * transform b points in standard pose*
    end
    P_current_frame ←computeMLS(P_current_frame)
    P_current_frame ←computeVoxeling(P_current_frame)
    for m ← 1 to n do
        fitness_scores ← computeReIdentificationScores(test_model, M_m)
        * fill row current_frame in the classification matrix C_m, with sum of current
        fitness_scores and the last K-1 rows previously inserted in the matrix of the classic
        algorithm*
    end
end
```

5.3 Ranking Based PCM

Algorithm 6: RankingBased Point Cloud Matching algorithm

Data:

- Set of M models of n subjects freely to move
- A multi set containing n set of test frames everyone related to a subject free to move filmed by Kinect sensor.

Result: A set of n classification matrix $C = \{C_1, C_2, ..., C_n\}$ everyone with N_k rows and n columns where k is the index of the model considered and N_k is the number of test frames representing subject k

```
for * current_frame in frames of current_model * do
    P_current_frame ← * point_cloud of current frame *
    * Divide the points of P_current_frame in body parts*
    for b in * body parts of P_current_frame* do
        * transform b points in standard pose*
    end
    P_current_frame ←computeMLS(P_current_frame)
    P_current_frame ←computeVoxeling(P_current_frame)
    for m ← 1 to n do
        fitness_scores ← computeReIdentificationScores(test_model, M_m)
        ranking ← computeRanking(fitness_scores)
        * fill row current_frame of the ranking matrix K_m with ranking *
        * fill row current_frame of the classification matrix C_m with sum of the last K rows
        of K_m matrix *
    end
end
```

The Ranking-based Point Cloud Matching (Algorithm 6) is an algorithm that implements information fusion at the decision-level like the algorithm in Section 5.2. The idea is the same of the previous algorithm, but instead of using fitness scores as weights, we use only the order of importance of the fitness scores.

We give an increasing ranking to a set of n fitness scores ordered in decreasing way, because the similarity is inversely proportional with these scores. With this method the best fitness score the current frame has reached, receive the lowest rank, the second best fitness score receive the second lowest rank and so on. Thus, we are weighting the models with a rank in order of similarity, losing the information about fitness score.

After the computation and the assignment of this ranking, the algorithm proceeds exactly as the LastK-PCM algorithm (see 5.2), summing the last K rows of the ranking matrix. This algorithm has two parameters:

- *ranking vector*: A set of n integers in increasing order where n is the number of gallery models. This represents the weight assigned to every position. In our test, we have used a linear ranking: $1, 2, ..., n$.

- K: number of frames to sum

Chapter 6

Experiments

In this section we will summarize the results we obtained with the novel techniques we introduced. All the algorithms have been tested on two public datasets, as explained in Section 6.1. We will give also a comparative table with the classification performance for all the algorithms tried. For measuring classification performance we have used the *rank1* and *Cumulative Matching Characteristic Curves (CMC)* which are widely used for evaluating re-identification algorithms ([15], [3]).

In order to use this measurements, the classification algorithm has to produce a kxn matrix where k is the number of valid frames and n is the number of models in the training set. Every row i of this matrix contains n floats that represent the similarity of the current frame i against the models. If the minimum value of this sequence is in the column j, ($j \leq n$) then model j is classified as the best similar to the frame i, if the column l ($l \leq n$) contains the second minimum than the model l is classified as the second best similar and so on.

rank-1 is defined as the percentage of frames classified with the best model i (model i has the lowest fitness score among gallery models), given that the correct model of the frame was i.

The *Cumulative Matching Characteristic Curve* is a way to visualize different ranks in a coordinate system. The measurement we used about it is the $nAUC$ which is the area behind it. The $nAUC$ can be expressed with the following formula:

$$nAUC = \frac{\sum_{i=1}^{n} rank_i}{number\ of\ frames} \tag{6.1}$$

where $rank_i$ is the number of test frames whose correct model is classified in the first i fitness scores.

6.1 Dataset

The datasets we have used for long-term re-identification are all public. We report here some information about them:

- **IAS-Lab RGBD-ID Dataset**[1]: This dataset is made of 22 tracks of 11 different people. It is a collection of frames recorded with the NiTE middleware. For every frame there are the following files:
  - rgb image(640x480 pixels)
  - depth image(640x480 pixels)
  - user map(640x480 pixels)
  - txt file with skeleton tracker joints position and links orientation (estimated with NiTE middleware)

  For every subject there are three sequences:
  - **Training**: With these frames we construct the 3D model of every subject.
  - **TestingA**: In these frames every subject has different clothes than in its **Training** sequence.
  - **TestingB**: In these frames every subject has the same clothes he had in **Training**, but the sequence is recorded in a different room.
- **BIWI RGBD-ID Dataset**[2]: The BIWI RGBD-ID Dataset is a RGB-D dataset of people targeted to long-term people re-identification from RGB-D cameras.

  It contains 50 training and 56 testing sequences of 50 different people. The dataset includes:
  - synchronized RGB images (captured with the highest resolution possible with a Microsoft Kinect for Windows i.e. 1280x960 pixels)
  - depth images
  - segmentation maps

[1] http://robotics.dei.unipd.it/reid/index.php/8-dataset/5-overview-iaslab
[2] http://robotics.dei.unipd.it/reid/index.php/8-dataset/2-overview-biwi

– skeletal data (as provided by Microsoft Kinect SDK)

In addition to this, for every person information about ground plane coefficients are available.

The dataset is divided in three parts:

– **Training**: In these sequences, people performs a certain route and motions in front of the Kinect sensor. These sequences are targeted to build 3D models of people.

– **Still**: In these sequences 28 people out of 50 present in the training phase are recorded in still position. Every person is still or slightly moving in place and most of them are dressed differently with respect to the training video.

– **Walking**: In these sequences the same people of the Still records are recorded while they are walking in the scene. They perform two walks frontally and two diagonally with respect to the Kinect.

6.2 PCM performance analysis

The first test we have done in our work is the analysis of the classification performance of the Point Cloud Matching algorithm with some new sequences. These sequences contain people already recorded for the IAS-Lab RGBD-ID Dataset (see Section 6.1) two years after its recording. The people have been recorded while rotating on theirself from -60 degrees to 60 degrees with respect to the Kinect sensor as reported in Figure 6.2. This orientation angle has been computed converting to euler angles the quaternion of the hip joint relative to the Kinect.

A sequence we have analysed is about the model 4 of the IAS-Lab RGBD-ID dataset. We have collected about 600 frames of him at various angles with respect to the Kinect sensor. The total *rank-1* gained by the PCM algorithm, with this sequence, was 43.44% and the nAUC was 89.99%, confirming that the algorithm still works fine even after two years. In Figure 6.2, is reported the fitness scores gained by PCM algorithm with the new sequence recorded and in Figure 6.2 is reported the nAUC with the same sequence. The model 4 fitness scores with the frames recorded are highlighted in black. The curves in the figure show that the best classification angle of a person for the PCM algorithm is from -20 to 20 degrees. The more a person varies his angle with respect to the Kinect, the more the performance of the algorithm degrades quickly even if the model is correctly recognized.

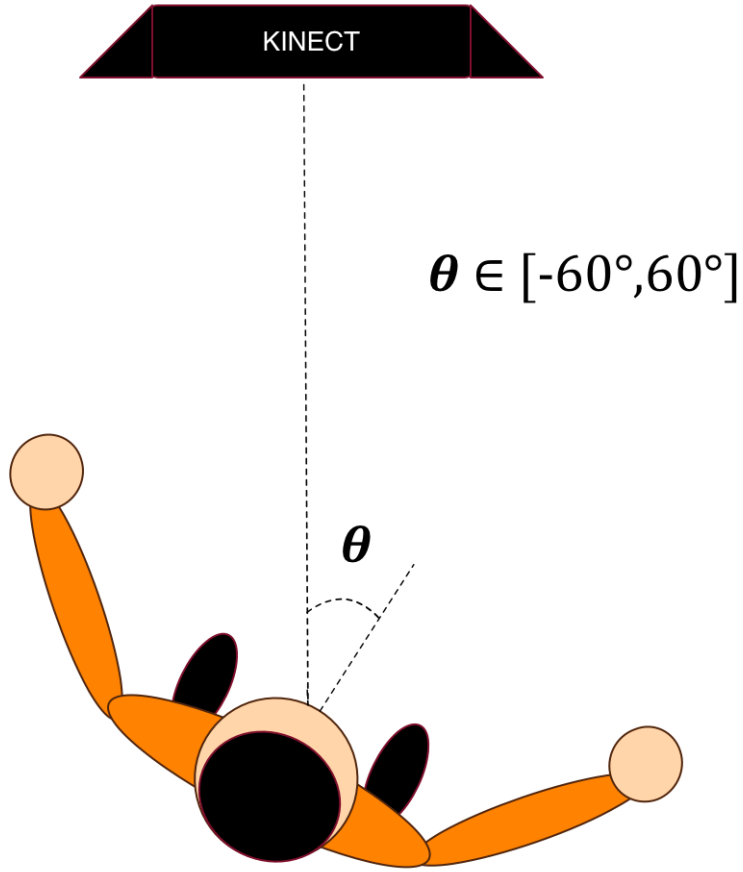


Figure 6.1: Orientation angles assumed by the person in the new sequence recorded. The person is in still position while he varies its angle with respect to the Kinect sensor

6.3 Multi-frame PCM

The Multi-frame Point Cloud Matching algorithm (Algorithm 4) consists of a multi-frame test model built online with test frames. This model is then matched with gallery models obtaining fitness scores. This algorithm has two parameters: the maximum number of frames in the test-model and the maximum fitness score that can have a single test frame matched with the current non-empty test-model. The latter parameter has been set to 0.1 while we have done a grid-search tuning of the number of frames in the test-model, using the datasets described in Section 6.1. In Tables 6.1, 6.2, 6.3, and 6.4 we report the results we have reached with the best parameters on the datasets.

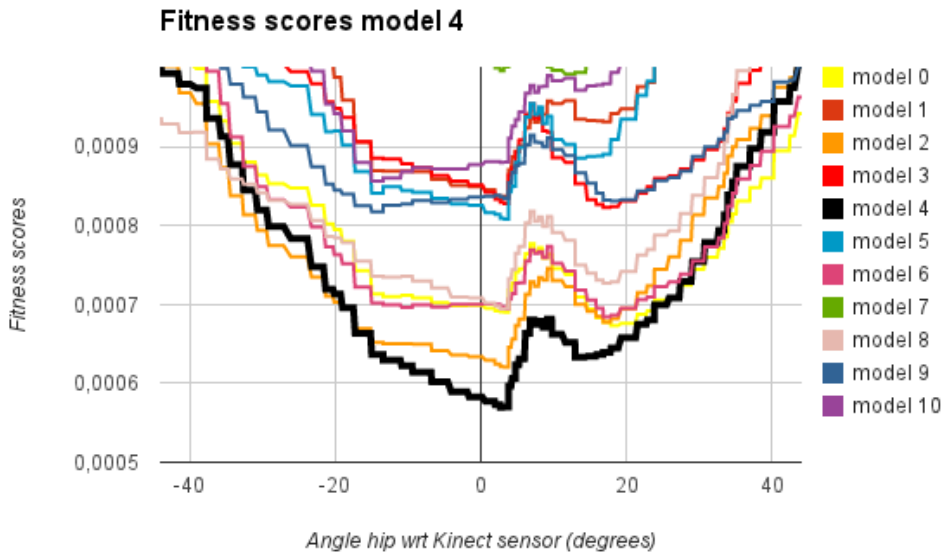


Figure 6.2: Fitness scores obtained with the PCM Algorithm with a new test sequence of model 4 recorded two years after the model has been built

In these tables our results are reported in white background cells while the ones obtained in the reference article[15] are visible in grey background cells.

TESTING A IAS-Lab RGBD-ID DATASET	*Reference Results:*	*28,60%*	*73,20%*
FRAMES IN THE MODEL		**rank1**	**nAUC**
2		31,15%	74,79%
4		**32,27%**	**75,54%**
5		31,45%	75,53%
6		31,67%	75,30%
8		31,80%	74,78%

Table 6.1: Multi-frame PCM Algorithm results on TestingA (IAS-Lab RGBD-ID dataset)

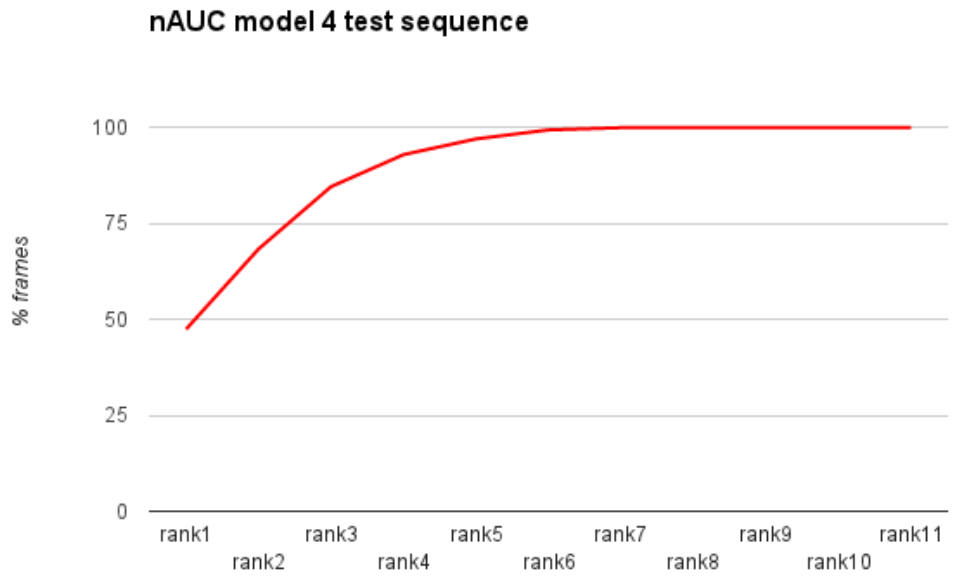


Figure 6.3: Cumulative Matching Characteristic Curve with the new test sequence of model 4

TESTING B IAS-Lab RGBD-ID DATASET	*Reference Results:*	*43,70%*	*81,70%*
FRAMES IN THE MODEL		**rank1**	**nAUC**
2		41,29%	80,33%
4		44,59%	81,52%
5		46,02%	81,68%
6		46,74%	81,91%
8		**49,52%**	**81,40%**

Table 6.2: Multi-frame PCM Algorithm results on TestingB (IAS-Lab RGBD-ID dataset)

STILL BIWI RGBD-ID DATASET	*Reference Results:*	*32,50%*	*89,00%*
FRAMES IN THE MODEL		**rank1**	**nAUC**
2		29,06%	83,97%
4		**30,95%**	**81,80%**
5		30,47%	81,67%
6		27,12%	81,35%
8		25,66%	80,93%

Table 6.3: Multi-frame PCM Algorithm results on Still (BIWI RGBD-ID dataset)

WALKING BIWI RGBD-ID DATASET	*Reference Results:*	*22,40%*	*81,60%*
FRAMES IN THE MODEL		**rank1**	**nAUC**
2		**24,26%**	**81,11%**
4		18,70%	77,16%
5		17,62%	77,60%
6		15,61%	76,80%
8		17,39%	76,82%

Table 6.4: Multi-frame PCM Algorithm results on Walking (BIWI RGBD-ID dataset)

This method improves the results for the IAS-Lab RGBD-ID dataset, but it performs worse in the BIWI RGBD-ID dataset. This is probably due to the noise in the test model. Even if the current frame is registered with the model before its inclusion in it, the noise in Kinect depth estimation affects the fitness scores. The results are more sensitive to noise in the BIWI dataset, because there are a considerable number of similar people recorded instead of the 11 in the IAS-Lab RGBD-ID.

With new sensors like Kinect v2, we think that this algorithm can perform extremely better. From the first tests we have done, in fact, the point clouds recorded with this new sensor seem to be less noisy and the contribution of every frame to the test model should be more positive.

6.4 LastK Point Cloud Matching

The LastK PCM algorithm is based on the sum of the last K fitness scores. It has only one parameter: the number of frames we consider in the total sum for every row (K). After the tuning phase of this parameter, we have obtained the results reported in Tables 6.5, 6.6, 6.7 and 6.8.

TESTING A IAS-Lab RGBD-ID DATASET	*Reference Results:*	*28,60%*	*73,20%*
K		**rank1**	**nAUC**
25		35,24%	73,82%
50		40,01%	74,80%
75		42,01%	75,46%
100		**45,92%**	**75,95%**

Table 6.5: LastK PCM Algorithm results on TestingA (IAS-Lab RGBD-ID dataset)

TESTING B IAS-Lab RGBD-ID DATASET	*Reference Results:*	*43,70%*	*81,70%*
K		**rank1**	**nAUC**
25		47,45%	80,35%
50		49,96%	81,01%
75		51,41%	81,43%
100		**51,71%**	**81,83%**

Table 6.6: LastK PCM Algorithm results on TestingB (IAS-Lab RGBD-ID dataset)

STILL BIWI RGBD-ID DATASET	*Reference Results:*	*32,50%*	*89,00%*
K		**rank1**	**nAUC**
25		**32,59%**	**85,76%**
50		30,40%	85,63%
75		30,97%	85,71%
100		30,30%	85,77%

Table 6.7: LastK PCM Algorithm results on Still (BIWI RGBD-ID dataset)

WALKING BIWI RGBD-ID DATASET	*Reference Results:*	*22,40%*	*81,60%*
K		**rank1**	**nAUC**
25		25,89%	81,89%
50		26,77%	81,35%
75		**26,77%**	**81,39%**
100		26,77%	81,39%

Table 6.8: LastK PCM Algorithm results on Walking (BIWI RGBD-ID dataset)

The results we have obtained with this algorithm show a better improvement in IAS-Lab RGB-ID Dataset. In the more challenging BIWI RGBD-ID dataset, the results are similar to those of the standard PCM algorithm. As for the Multi-frame algorithm results (see Section 6.3), also in this case we are giving too importance to the noise. In fact, considering fitness scores sums we consider also frames with bad scores, compromising the positive contribution that the history gives. Thus, this algorithm can be improved with a quality-control check. One interesting way is to check if the mean fitness score of a frame significantly improves the variance of the fitness scores if it is added in the sum.

6.5 Ranking-Based Point Cloud Matching

The Ranking-Based Point Cloud Matching algorithm addresses the people re-identification problem like the LastK PCM algorithm (see Section 6.4). The difference is that LastK takes into account the fitness scores of the last K frames considering noisy frames like the others. On the contrary, Raking-Based PCM algorithm, gives a fixed ranking to the fitness scores of a frame ordered by similarity on the gallery models. In this way a noisy frame does not change so much the variance of the sum as it happens in the LastK PCM algorithm. With this algorithm there are two parameters: the ranking list of integers and the number of frames considered in the sum (K). In our work we have used the unit list $(1, 2, ..., n)$ as the ranking list and we have tuned K with it. The results with the datasets are reported in Tables 6.9, 6.10, 6.11 and 6.12.

TESTING A IAS-Lab RGBD-ID DATASET	*Reference Results:*	*28,60%*	*73,20%*
K		**rank1**	**nAUC**
25		33.85%	76.15%
50		34.39%	76.46%
75		**35.53%**	**76.63%**
100		34.41%	76.66%

Table 6.9: Ranking-Based Algorithm results on TestingA (IAS-Lab RGBD-ID dataset)

TESTING B IAS-Lab RGBD-ID DATASET	*Reference Results:*	*43,70%*	*81,70%*
K		**rank1**	**nAUC**
25		49.89%	84.53%
50		52.41%	86.41%
75		**55.20%**	**87.88%**
100		55.97%	88.53%

Table 6.10: Ranking-Based Algorithm results on TestingB (IAS-Lab RGBD-ID dataset)

STILL BIWI RGBD-ID DATASET	*Reference Results:*	*32,50%*	*89,00%*
K		**rank1**	**nAUC**
25		33.61%	91.22%
50		33.65%	91.06%
75		**34.88%**	**91.09%**
100		34.78%	91.15%

Table 6.11: Ranking-Based Algorithm results on Still (BIWI RGBD-ID dataset)

WALKING BIWI RGBD-ID DATASET	*Reference Results:*	*22,40%*	*81,60%*
K		**rank1**	**nAUC**
25		27.60%	88.65%
50		29.04%	88.84%
75		**29.04%**	**88.89%**
100		29.04%	88.89%

Table 6.12: Ranking-Based Algorithm results on Walking (BIWI RGBD-ID dataset)

The results we have obtained with this algorithm improve all the reference article results on both datasets. With the ranking, we achieve the goal to give less importance to the noisy frames. However, we have less improvement in some cases like TestingA with respect to the previous algorithm (Section 6.4). With this algorithm we give the same importance to all the test frames and, with a method that detects noisy frames, this algorithm can be overtaken by other algorithms like LastK Point Cloud Matching.

6.6 Future works

The results presented in this work show a general performance improvement in rank-1 and nAUC with regard to the previous work [15]. There are some possible improvements that can be done:

- **Introduce Kinect v2**: In this work we have used Microsoft Kinect sensor which uses a structured light projection algorithm to capture depth information as explained in Section 3.3. From some experiments we have done, Kinect v2 sensor can give more accurate and less-noisy depth information. Running the algorithms we have proposed in new datasets recorded with this sensor should be sufficient to obtain good results.

- **Introduce a way to detect noisy frames**: As described before (Section 6.5), if we detect noisy frame we can improve the PCM algorithm classification performance and the multi-frame techniques we have introduced. A way to do this is to consider the current fitness scores with the variance of the fitness scores previously seen. Like described in Section 6.2 in fact, the orientation of a person considerably affects these scores and can give a precious information on a bad classification.
  Another idea, if the re-identification scene is always the same, is to train a classifier to predict if a frame is noisy or not.

- **Combine different algorithms together**: Every algorithm we have introduced show an improvement in the classification performance with regard to the classic one even if only with one dataset. An interesting idea is to fuse these algorithms together, with a set of weights. In this way, we will combine different decisions overtaking problems of every single algorithm.

Chapter 7

Conclusions

In this work we introduced three new algorithms for long-term re-identification based on the Point Cloud Matching algorithm [15]. These algorithms exploit 3D shape information obtained by consumer depth sensors like Microsoft Kinect. Moreover, they are also efficient, in order to be executed in real time on-board of mobile robots.

The Multi-frame Point Cloud Matching algorithm is a method that consists in building a person model online also at test time with the last K frames acquired. Later this model is used instead of the current frame for computing the fitness scores with the training models.

The LastK Point Cloud Matching algorithm performs classification by exploiting the sum of the fitness scores of the last K frames computed with the classic Point Cloud Matching algorithm.

The Ranking-Based Point Cloud Matching algorithm uses a similar concept of the LastK PCM algorithm, but instead of fitness scores, it computes a ranking of the set of similarity scores of a frame.

These algorithms was tested with two public datasets: IAS-Lab RGBD-ID and BIWI RGBD-ID.

The results we reached are better than the PCM algorithm[15] in terms of rank-1 and nAUC which are indices commonly used in state-of-the-art algorithms for long-term re-identification.

With Multi-frame Point Cloud Matching we obtained good results with the IAS-Lab RGBD-ID dataset, but weaker results with the BIWI RGBD-ID dataset. This is probably due to the lack of methods for cleaning the test model from noisy frames.

The best method we introduced is the Ranking-Based Point Cloud Matching algorithm. This algorithm always improves state-of-the-art classification performance on both the datasets we used, but shows less improvement in some cases with respect to the other algorithms we introduced. This problem

is due to the fact that noisy and good frames are equally weighted. There are currently no methods to establish whether a frame is good or noisy.

With the last algorithm we proposed, the LastK Point Cloud Matching algorithm, we obtained the better improvements in some cases, but it performs worse than the standard PCM algorithm in others. This is due to noisy frames fitness scores that add large variance to the total sum of scores.

In this work, we also introduced the concept of *Silhouette*, which is a signature of the side view of a person's cloud. This method is computationally less expensive than standard re-identification algorithms, but we haven't obtained good results with them. This is probably due to the Kinect imprecision at every frame. With the advent of more accurate sensors, the algorithm based on silhouettes will probably perform better.

Finally, we studied how fitness scores change when varying the orientation angle of a person with the standard PCM algorithm. We noticed that the PCM algorithm performs better when a person is still and that even when people are seen after two years, it obtains good classification performance.

We think that with the advent of new sensors like Kinect v2 which use different, more accurate algorithms to obtain depth information with respect to Kinect 1, the methods we presented will obtain further improvements.

References

[1] Davide Baltieri, Roberto Vezzani, and Rita Cucchiara, *Sarc3d: a new 3d body model for people tracking and re-identification*, Proceedings of the 16th International Conference on Image Analysis and Processing (Ravenna, Italy), September 2011, pp. 197–206.

[2] B. I. Barbosa, M. Cristani, A. Del Bue, L. Bazzani, and V. Murino, *Re-identification with rgb-d sensors*, First International Workshop on Re-Identification, October 2012.

[3] Alberto Basso, *Trasformazione a posa standard di point cloud di persone per re-identificazione*, Master's thesis, Università degli studi di Padova, Padova, 2012/2013.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, *Speeded-up robust features (surf)*, Comput. Vis. Image Underst. **110** (2008), no. 3, 346–359.

[5] P.J. Besl and Neil D. McKay, *A method for registration of 3-d shapes*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **14** (1992), no. 2, 239–256.

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, *Brief: Binary robust independent elementary features*, Proceedings of the 11th European Conference on Computer Vision: Part IV (Berlin, Heidelberg), ECCV'10, Springer-Verlag, 2010, pp. 778–792.

[7] Michela Farenzena, Loris Bazzani, Alessandro Perina, Vittorio Murino, and Marco Cristani, *Person re-identification by symmetry-driven accumulation of local features*, Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010) (San Francisco, CA, USA), IEEE Computer Society, 2010.

[8] F. Fleuret, H. Ben Shitrit, and P. Fua, *Re-identification for improved people tracking*, Person Re-Identification (S. Gong, M. Cristani, Y. Shuicheng, and C. C. Loy, eds.), Springer, 2014, pp. 311–336.

[9] H. Grabner and H. Bischof, *On-line boosting and vision*, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 1, June 2006, pp. 260–267.

[10] Anil Jain, Lin Hong, and Ruud Bolle, *On-line fingerprint verification*, IEEE Trans. Pattern Anal. Mach. Intell. **19** (1997), no. 4, 302–314.

[11] Kourosh Khoshelham and Sander Oude Elberink, *Accuracy and resolution of kinect depth data for indoor mapping applications*, Sensors 2012, 12, 1437–1454. 2013, p. 8238.

[12] David Levin, *The approximation power of moving least-squares*, Math. Comput. **67** (1998), no. 224, 1517–1531.

[13] D.G. Lowe, *Object recognition from local scale-invariant features*, Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, 1999, pp. 1150–1157 vol.2.

[14] Matthias Luber, Luciano Spinello, and Kai O. Arras, *People tracking in rgb-d data with on-line boosted target models*, In IEEE/RSJ Int. Conf. on, 2011.

[15] Matteo Munaro, Alberto Basso, Andrea Fossati, Luc Van Gool, and Emanuele Menegatti, *3d reconstruction of freely moving persons for re-identification with a depth sensor*.

[16] Matteo Munaro, Stefano Ghidoni, Deniz Tartaro Dizmen, and Emanuele Menegatti, *A Feature-based Approach to People Re-Identification using Skeleton Keypoints*, IEEE International Conference on Robotics and Automation (ICRA2014), 2014.

[17] Matteo Munaro and Emanuele Menegatti, *Fast RGB-D People Tracking for Service Robots*, Autonomous Robots (2014).

[18] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng, *Ros: an open-source robot operating system*, ICRA Workshop on Open Source Software, 2009.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, *Orb: An efficient alternative to sift or surf*, Computer Vision (ICCV), 2011 IEEE International Conference on, Nov 2011, pp. 2564–2571.

[20] Radu Bogdan Rusu and Steve Cousins, *3D is here: Point Cloud Library (PCL)*, IEEE International Conference on Robotics and Automation (ICRA) (Shanghai, China), May 9-13 2011.

[21] R.B. Rusu, N. Blodow, and M. Beetz, *Fast point feature histograms (fpfh) for 3d registration*, Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, May 2009, pp. 3212–3217.

[22] R.B. Rusu, N. Blodow, Z.C. Marton, and M. Beetz, *Aligning point cloud views using persistent feature histograms*, Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, Sept 2008, pp. 3384–3391.

[23] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, *Real-time human pose recognition in parts from a single depth image*, CVPR, IEEE, June 2011.

[24] Federico Tombari, Samuele Salti, and Luigi Di Stefano, *Unique signatures of histograms for local surface description*, Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III (Berlin, Heidelberg), ECCV'10, Springer-Verlag, 2010, pp. 356–369.

[25] Federico Tombari, Samuele Salti, and Luigi di Stefano, *A combined texture-shape descriptor for enhanced 3d feature matching.*, ICIP, IEEE, 2011, pp. 809–812.

[26] Paul Viola and Michael Jones, *Rapid object detection using a boosted cascade of simple features*, 2001, pp. 511–518.

[27] R.P. Wildes, *Iris recognition: an emerging biometric technology*, Proceedings of the IEEE **85** (1997), no. 9, 1348–1363.

Ringraziamenti

Questo lavoro segna la fine del mio percorso universitario. Ci tengo a ringraziare tutte le persone che mi sono state vicine e che mi hanno aiutato e sostenuto in questi anni.

Ringrazio molto la mia famiglia che mi ha dato la possibilità economica e il sostegno morale necessario a terminare questo percorso. Senza il loro aiuto questo lavoro non esisterebbe e difficilmente riuscirò a ripagarli equamente.

Ringrazio anche Silvia che ha avuto molta pazienza con me, sopportando la mia visione spesso troppo ingegneristica del mondo. Questo lavoro è dedicato a lei.

Un grazie speciale ai miei compagni universitari che hanno reso più semplice il mio percorso grazie alle nostre sedute di *brainstorming* e alle battute per sdrammatizzare ogni evento.

Un ringraziamento particolare anche allo IAS-LAB, al mio relatore prof. Emanuele Menegatti e al mio correlatore Dott. Ing. Matteo Munaro, che sono stati sempre disponibili fornendomi materiale e spiegazioni nonostante le mie continue richieste avrebbero fatto scappare chiunque altro.

Infine ringrazio chiunque legga questa tesi ricordando che tutti gli errori che vi sono contenuti sono da imputare a me.