J. Vis. Commun. Image R. 25 (2014) 39–52

Contents lists available at SciVerse ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

An adaptable system for RGB-D based human body detection and pose estimation

Koen Buys [a,*], Cedric Cagniart [b], Anatoly Baksheev [c], Tinne De Laet [a], Joris De Schutter [a], Caroline Pantofaru [d]

[a] University of Leuven, Dept. Mech. Eng., Celestijnenlaan 300, 3001 Heverlee, Belgium
[b] Technical University of Munich, Boltzmannstr. 3, 85748 Garching b., München, Germany
[c] Itseez, 603000 N. Novgorod, Korolenko 19b, Russia
[d] Willow Garage, Willow Road 68, Menlo Park, CA, USA

ARTICLE INFO

Article history:
Available online 29 March 2013

Keywords:
Real-time
Body part recognition
Joint locations
Pose detection
RGB-D data
Person detection
Random decision forest
Open source
Motion capture

ABSTRACT

Human body detection and pose estimation is useful for a wide variety of applications and environments. Therefore a human body detection and pose estimation system must be adaptable and customizable. This paper presents such a system that extracts skeletons from RGB-D sensor data. The system adapts on-line to difficult unstructured scenes taken from a moving camera (since it does not require background subtraction) and benefits from using both color and depth data. It is customizable by virtue of requiring less training data, having a clearly described training method, and a customizable human kinematic model. Results show successful application to data from a moving camera in cluttered indoor environments. This system is open-source, encouraging reuse, comparison, and future research.

1. Introduction

Detecting people, estimating body poses, and tracking them is an important problem that arises in diverse applications. Known as 'skeleton tracking' or 'markerless motion capture', it can be found as pedestrian blob tracking in automotive applications, body part detection in image search, stick figure tracking for gaming, and all the way to tracking fully articulated models for biomechanics. For personal robots that are designed to interact with people, body pose estimation can enable learning from demonstration, object hand-off, human-robot collaboration, and more natural interfaces. The wide range of applications makes human detection and body pose estimation an active field, and the recent introduction of affordable color and depth (RGB-D) cameras has accelerated the progress in this field.

Given the range of applications, a human detection and pose estimation systems needs to adapt to many situations. Different tasks require different estimation granularity (from bounding boxes to fully deformable models), as well as different levels of accuracy for different body parts. Variations also occur in environmental conditions; indoor service robotics applications, for example, often require that a tracker cope with a moving camera, as well as clutter, occlusions, and changing lighting, as in Fig. 1. It is unlikely that a single tracker will successfully deal with all variations, making this an open area of research.

In this paper, we present a customizable and adaptable system for skeleton tracking that both enables retraining for different applications and adapts on-line to its environment. We apply the system to a personal robotics scenario: a moving camera in cluttered indoor environments.

The most popular approach to skeleton tracking is by Shotton et al. [1] and the XBOX team [2], and must be used as a black-box with the Kinect. This approach segments a person from the background, and then uses depth features to label each pixel on the foreground with a body part label, from which a skeleton is extracted. This approach works well for its intended use: a large living room with little occlusion where the human is facing camera: the 'fixed-camera-living-room scenario'. However, it relies on background subtraction, expensive training, and the code is closed source, making it difficult to use for other applications or in other scenarios. Furthermore, this method only relies on depth data, discarding image data, and does not adapt on-line to new environments.

While taking inspiration from Shotton et al. our system facilitates research for applications other than the fixed-camera-living-room-scenario. The contributions of our system are as follows.

As a first contribution, our approach does not require pre-processing by background subtraction nor building a map of the envi-

* Corresponding author.
E-mail address: buys.koen@gmail.com (K. Buys).

1047-3203/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jvcir.2013.03.011

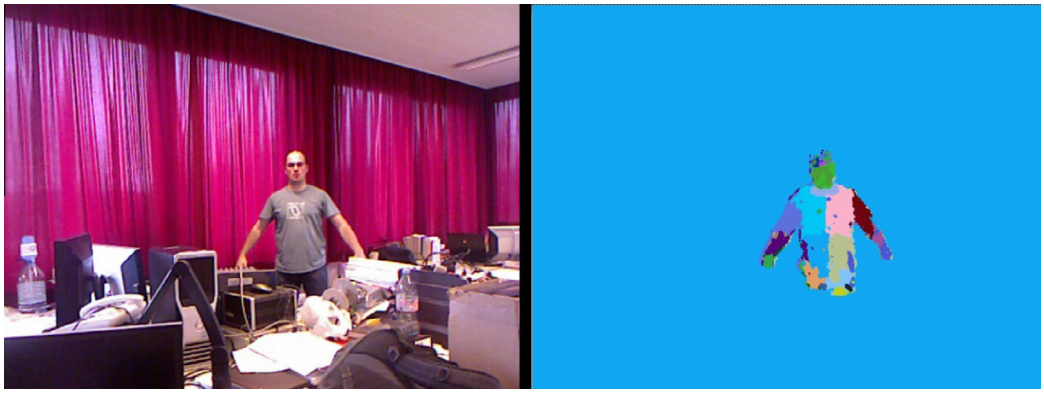

Fig. 1. A difficult person detection and tracking scene with occlusions and clutter. The left subimage shows the original scene. The right subimage shows the resulting body part labelings, with each color a different part. The complete skeleton is not visible and yet the visible body parts are found. (Best viewed in color)

ronment. Our approach evaluates every pixel[1] in an image as a candidate body part, and skeletons are extracted from all of the part hypotheses. Robustness and precision are obtained in two ways. First, our approach relies on an *underlying kinematic model*. Second, our approach uses an additional iteration of the algorithm that segments the body from the background. From the initial iteration, a rough skeleton is extracted, which is then used to *estimate an appearance model online*, and apply that model to perform foreground-background segmentation. Thanks to this second step, our system does not rely on background subtraction and can therefore be applied to a moving camera and can detect both moving and stationary people. Furthermore, there is no need to build a point cloud map of the environment, which (despite advances in point cloud alignment [3]) is still difficult for room-sized point clouds, moderate camera motion, and narrow field-of-view RGB-D cameras. The proposed approach furthermore detects humans and estimates body poses among clutter and with body part occlusions thanks to the search for all possible body part hypotheses and the underlying kinematic model. Additionally, no unnatural initialization poses are required.

As a second contribution, our approach adapts on-line to the user's clothing and environment via an online estimated appearance model combining color with depth-based labeling. Few other systems combine color and depth. As in [1], an initial pixel labeling is performed using a random decision forest (RDF) [4–7] based on depth features, and a rough human skeleton is extracted. Using only depth features requires an extremely large training dataset [1], which is expensive to create. Instead, we use this labeling as a rough estimate to bootstrap more robust labeling based on both color and depth, increasing performance, robustness and precision, while decreasing dependency on training data. Since it requires less training data, the system is easier to retrain.

As a third contribution, we present a method for RDF training, including data generation and cluster-based learning, that enables classifier retraining for different body models, kinematic models, poses or scenes. This allows future users of our system to customize the training to their own needs.

Finally, a key contribution is that our system, data, and dependencies are completely open source, to use, compare, or improve our system, boosting research in the field.

[1] Remark that since we are considering a RGB-D image, a pixel is also connected with a depth measurement. Therefore the 'pixel' directly corresponds to a voxel. When refering to a pixel in this paper, we are actually refering the corresponding 3D voxel.

2. Related work

Interest in markerless motion capture has existed for a long time. Moeslund et al. [8,9] and Poppe [10] provide surveys of the literature up to their time.

There are many image-based approaches [11–20], etc. While the idea of using only images is enticing, 2D approaches are plagued by the difficulty of separating subjects from backgrounds, also known as the detection or figure-ground segmentation step. Thus, they either suffer from poor performance, or need uniform backgrounds, or need stationary cameras with static backgrounds for background subtraction. Silhouette-based methods additionally suffer from pose ambiguities.

A growing trend is to use 3D information from multiple cameras, time-of-flight cameras, or RGB-D sensors like the Kinect. Two issues in state-of-the-art approaches are initialization and background subtraction. Many methods require initialization during which the user has to take specific and unnatural pose to calibrate the model [21]. Furthermore, many methods require that the human body is segmented from the image or rely on background subtraction [22] assuming fixed camera and static background. Some depth-based methods estimate body extrema (such as hands) [23,24], which can work well when those extrema are visible. Many methods are part-based, such as [21,25]. In the extreme, a full 3D mesh is created [26]. The body estimates can be noisy, so a refinement step like matching to a pose database [27] is useful.

Most methods work either in 2D image space or 3D depth, few combine the two. In contrast, our algorithm's robustness is due to the combination of color and depth information.

RGB-D cameras have accelerated skeleton tracking research with approaches such as that of Shotton et al. [28,1] and the XBOX team [2]. Shotton et al. used a randomized decision forests classifier with simple depth-based features [4,1] to create a real-time system. A downside is that it requires a large training set of labeled depth images of different people in different poses. Data was created by capturing users' movements and then rendering depth images with a graphics model, as in [29,30]. This dataset has not been released. The Shotton et al. estimator has been tuned for a large living room with a stationary narrow field-of-view camera, and little user occlusion. There is little post-processing to recover from poor part localization by the classifier. Also, accurate person-background segmentation is assumed, which would be difficult from a moving camera among clutter. Given these issues, the Shotton et al. system is difficult to use for indoor robotics scenarios. To ensure that a single system is applicable to multiple environments, it needs to be customizable and adaptable: to learn

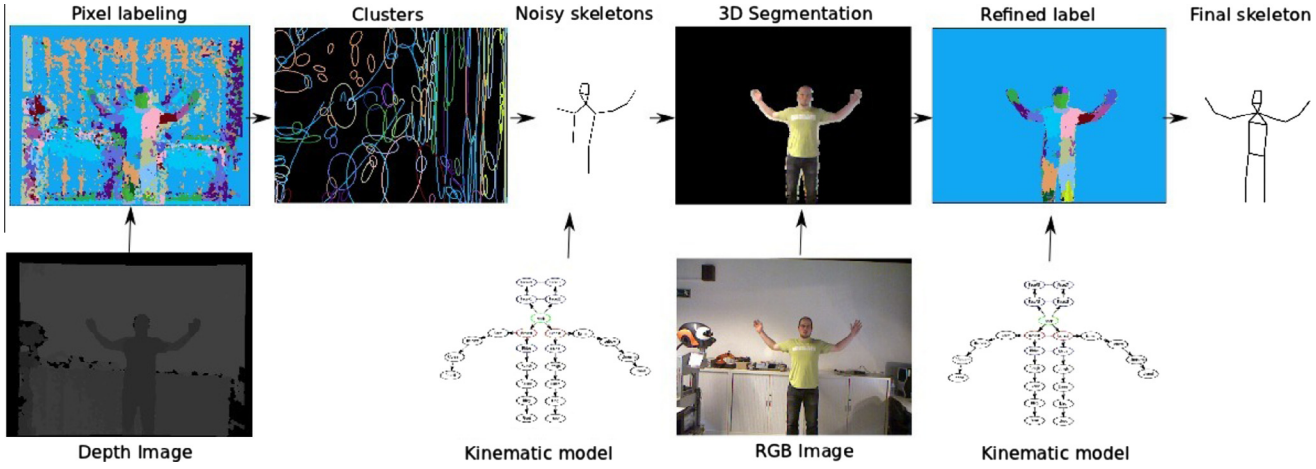

Fig. 2. Run-time system flow diagram. A depth image is used to perform a noisy pixel labeling over the entire image, which is then clustered into human body parts. From these clusters, a person is located and skeleton is extracted based on a kinematic model. This initial labeling and skeleton are used to bootstrap the on-line learning of a color and depth-based appearance model which then refines the pixel labeling and person segmentation, leading to more accurate and complete skeleton extraction. The output of the system includes a skeleton of the person, a part-based pixel labeling, and a segmentation of the person from the background.

about new environments, to be more modular, easier to train, and openly available for modification. In this paper, we present such an approach.

3. System overview

The run-time goal of our system is to receive data from an RGB-D sensor and output the 3D locations of the human body parts that are predefined in a kinematic model. At a high level, this process can be outlined as seen in Fig. 2. For a single depth map frame, every pixel is labeled as belonging to a single body part using a RDF approach similar to [1] and features like those in [4,1], in the pixel-wise body part labeling step (Section 4). Note that unlike [1,31], there is no background subtraction or fixed-pose initialization before the pixel labeling. The obtained initial pixel labels are very noisy. Therefore, the body-part proposal step (Section 5) smooths the labels and then clusters them into more robust part estimates. From the statistics of the part estimates, a search for feasible kinematic trees is conducted during the kinematic tree search step (Section 6), resulting in person detections and noisy skeletonization with missing body parts and poor localization.

To improve the obtained estimate, a second iteration of pixel-wise body part labeling, body-part proposal, and kinematic tree search is performed based on a new estimate of the pixels actually belonging to the person being detected. To obtain a new estimate of which pixels actually belong to the person, an appearance model is estimated online during an appearance model estimation for segmentation refinement step (Section 7). The initial noisy estimate obtained during the first iteration is used as a seed for color and depth-based segmentation that retrieves missing body parts and better localizes existing parts. This process can be used for multiple people among clutter and occlusion. By running a second iteration of pixel-wise body part labeling, body-part proposal, and kinematic tree search on the obtained image, a more robust and accurate human body pose estimate is obtained.

In the sections below, we describe each component of the system (in bold) in detail. In each section, we will highlight how our design choices lead to a customizable and adaptable system ready for re-use by other researchers.

4. Pixel-wise body part labeling

The pixel-wise body part labeling step uses an RDF classifier to assign body part labels to pixels. In this section, we will describe the RDF training and application procedures. Section 4.1 discusses the training data. Section 4.2 explains how this data is used to train an RDF classifier. Finally, Section 4.3 describes how the RDF is used to label the pixels.

4.1. Training data

To train the RDF used in the pixel-wise body part labeling, a corpus of training depth images annotated with parts is needed. Our system can cope with noisy pixel labels, which significantly reduces the amount of training data required to train the trees. However, even this reduced corpus of training data has significant size and would be expensive to collect from people performing motions in front of a depth camera. Instead, we tackle the problem of generating training data by mapping real motion capture data onto a virtual model of a person [29,30].

Below, we describe the data generation process outlined in Fig. 3 by elaborating on the human body model (Section 4.1.1), the pose information (Section 4.1.2), and the rendering of the training data (Section 4.1.3). This process is customizable by virtue of its modularity and reliance on only free open-source tools and data.

4.1.1. The human body model

The virtual model of a person used in our system is the MakeHuman (MH) model [32] shown in Fig. 3(b). The model consists of a mesh structure with an underlying kinematic chain that can be parametrically morphed (by age, height, gender, etc.) as illustrated in Fig. 4. Although this model was created for animated character creation rather than bio-mechanical applications, it is sufficiently accurate and flexible to model a large portion of the population [33], for most applications in personal robotics, entertainment, etc. Previous work [34,35] showed that matching this model onto a specific person can be done automatically, allowing for runtime customization.

4.1.2. Pose information

To articulate the human model into realistic poses, we use the CMU MoCap database [36] (Fig. 3(a)). This dataset contains a large range of human motions such as human interaction, sports, dancing, etc., captured using a Vicon system.

Two pre-processing steps are needed to make the data suitable. First, task-appropriate data must be selected. Since our application is detecting people during their daily activities in indoor office or home environments, we manually selected image sequences of

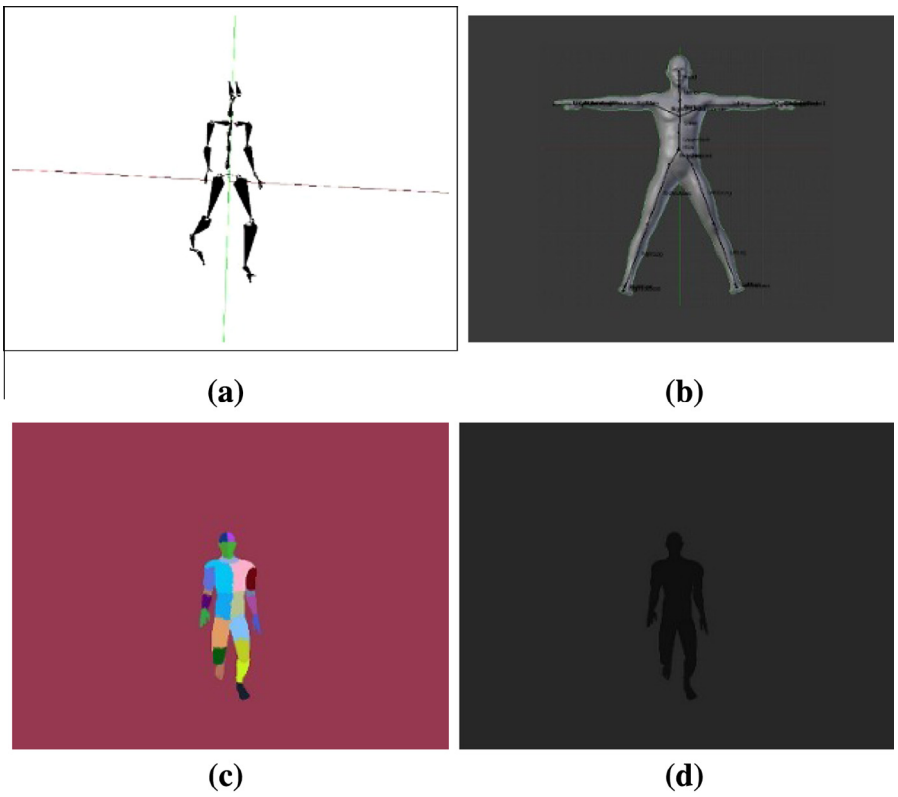

Fig. 3. Training data generation. (a) Motion Capture (MoCap) is mapped onto (b) a 3D graphics body model. With the addition of body part annotations, this pipeline produces (c) a body part-labeled model and (d) a depth image.

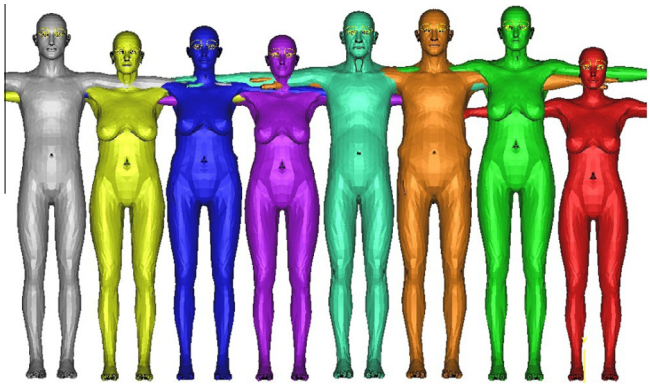

Fig. 4. MakeHuman allows anthropometric morphing to cover the population. (Best viewed in color)

every-day activities, and omitted extreme actions such as break-dancing. This reduced the dataset to 380,000 poses.

Second, the data must be sparsified to accelerate training and avoid biasing the trees with unbalanced data. The original data was collected at 120 Hz and contained many repeated poses. Greedy sparsification was performed resulting in a subset of the data containing 80,000 poses in which each pose is at least five degrees in joint angles from its nearest neighbor.

4.1.3. Rendering the training data

To render the training data, the kinematic chain from the CMU MoCap must be mapped onto the MH model. Since the former is less articulated than the latter, the mapping is a manual process, but it only needs to be performed once. The annotation of body parts onto the model is also a one-time manual process. We currently differentiate between 25 body parts, as seen in Fig. 3(c) (four for the head, neck, four for the torso, upper arms, elbows, lower arms, hands, upper legs, knees, lower legs and feet). Background pixels are set to a large depth value. Blender and OpenGL are used for skeleton mapping and rendering annotated depth images.

Future users may adapt this modular process to obtain training data that is better suited for the task at hand, perhaps by adding objects or refining the body. Future work will add a sensor-dependent noise model into the depth maps.

4.2. Learning random decision forest

This section describes how the training data (Section 4.1) is used to learn the RDF, including describing the features (Section 4.2.1) and and the creation of the RDF using MapReduce (Section 4.2.2).

4.2.1. Features and classifier

For pixel-wise labeling of the depth image, we use the features from [4,1]. For a pixel x in image I, a feature is the difference in depth between two other randomly chosen points:

$$f_\Theta(I,x) = d_I\left(x + \frac{o_1}{d_I(x)}\right) - d_I\left(x + \frac{o_2}{d_I(x)}\right) \quad (1)$$

where $d_I(x)$ is the depth of x, and $\Theta = (o_1, o_2)$ is a pair of pixel location offsets (which are normalized by depth). During training, feature values are computed for pixels on the body and discretized into 50 uniform bins (as [1]). Pixels recognized as belonging to the background are given a large depth.

For each pixel, the feature vector $\hat{f}_{\hat{\Theta}}(I,x)$ contains 2000 features generated by randomly sampling offset pairs. For training, this feature vector is calculated for 2000 randomly chosen pixels on each body. The list of offset pairs, $\hat{\Theta}$, is common for all pixels. The fea-

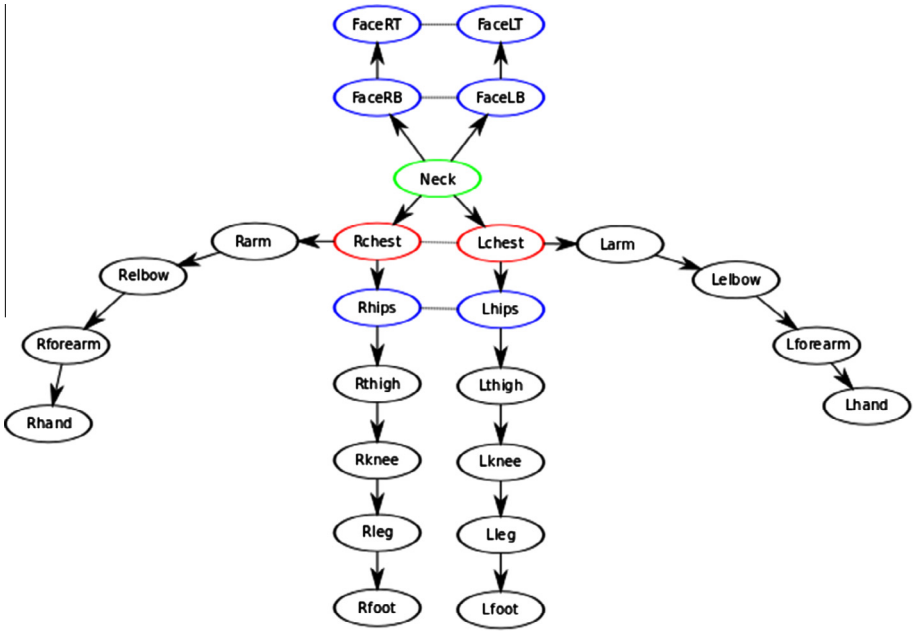

Fig. 5. The used kinematic model, this indicates the order in which parts are connected in parent–child pairs.

ture vectors with their ground truth labels, $\hat{f}_{\hat{\Theta}}(I,x,c)$, produce a data file of 300 GB for a single body model in 80 K poses.

From these feature vectors, a randomized decision forest is learned. Decision forests are effective [4,5] and can be implemented efficiently on a GPU. Each of the N_t trees in the forest produces a posterior distribution over the pixel label, $P_t(c|I,x)$, which are combined to give a forest estimate $P(c|I,x)$ as:

$$P(c|I,x) = \frac{1}{N_t}\sum_{t=1}^{N_t} P_t(c|I,x). \tag{2}$$

Every pixel in the image is labeled, including the background. Noise from background pixels, plus the tendency of decision forests to over-fit data, results in a noisy classification. An initial smoothing step is performed using a mode blur filter thresholded by depth. In practice, the filter has a 5×5 neighborhood and 300 mm depth threshold.

4.2.2. Creating an RDF with MapReduce

Random decision forest learning is efficiently performed in parallel by formulating it as a MapReduce [37] problem. MapReduce is ideal for tasks where the input data can be split up in shards without dependencies between them so that they can be processed concurrently. As its name indicates, it consists of a Map and a Reduce step. A *master node* will manage the entire process of cutting the data into parts, distributing it to be processed in parallel during the Map step, retrieving the results from the Map step and distributing again to be processed during the Reduce step. To explain the MapReduce algorithm we use a running example of counting the number of occurrences of certain words in a document.

The *Map step* consists of a function that is given a shard of the raw data by the master node and Maps this onto a set of $(key, value)$ pairs. In the running example each Map task is assigned a part of the document and creates a set of keys, being the words and a value indicating the number of occurrences in the given document part.

Afterwards the master node gathers the $(key, value)$ pairs from the Map step and gives pairs with the same key to a Reduce step. Therefore, the Reduce step is distributed according to the number of keys generated in the Map step.

The *Reduce step* consists of a function that reduces the set of $(key, value)$ pairs. The result of this reduction can be a list but is typically a single value. In the running example, each Reduce node is responsible for a particular key, i.e word in this case. It receives all the (word,word count) pairs, i.e. all the word counts of a particular word in different document parts, and sums all the word counts to produce an overall word count over the entire document.

The MapReduce problem formulation is very general. Therefore, when the RDF has to be relearned, the MapReduce can be easily executed again using the available computing clusters. For this paper, the MapReduce for learning the RDF was performed on an Apache Hadoop Cluster [38–40]. The power of Hadoop lies in that it keeps track of where each block of data is stored and executes the MapReduce steps on the compute node closest to the data. This increases performance and lowers network traffic.

Training a single decision tree occurs as follows [1]:

1. From the set of offset vectors $\hat{\Theta} = (o_1, o_2)$ and feature thresholds τ, propose a random set of splitting candidates Φ.
2. Compute the information gain from splitting the data at each Φ_i.
3. Compute the splitting candidate Φ_i^* that maximizes information gain.
4. If the gain is high, split the data and recurse on the two resulting subsets.

These steps are implemented as three MapReduce tasks:

- Find the split: The Map task allows each feature to vote for a split point. The Reduce task accumulates the votes into histograms and computes the information gain. The best split point is written to the Hadoop file system (HDFS).
- Split the data: The Map task uses the splitting threshold to map the feature vectors to odd or even values. The Reduce task then creates the two datasets, and a Cleanup task writes them to the file system.

- In memory: It is much more efficient to work in memory on a node once the data is sufficiently divided to fit. The Map task gathers all of the distributed files from the HDFS into a single file, and the Reduce task splits the data in memory.

Generating the training data and learning a decision tree with 16 levels using Hadoop took seven days for one body model in 80 K poses on a 16 node cluster. The 16 cluster nodes were eight-core Xeon X5570 systems with 48 GB RAM. The initial data generation took place on a single node with two GTX550ti GPUs.

By using the open infrastructure of MapReduce on a Hadoop platform, training can be conducted on non-homogeneous clusters and will automatically load balance, which allows our system to be retrained as necessary on all available hardware. In addition, training with larger datasets can be ported to a cloud cluster such as the publicly available Amazon Web Service cloud (which provides Elastic MapReduce (EMR) with Hadoop as a service.).

4.3. Using random decision forest

The output of the RDF learning is a number of decision trees. Each of these trees is 20 levels deep and thus contains little over a million leaf nodes. During run-time up to four of these trees are loaded into the GPU to form a RDF.

Each pixel is then passed through all of the N trees. At each tree node, a feature $f_\Theta(I,x)$ is compared against a threshold value learned during training. At the leaf nodes each of the N_t trees contains a probability distribution $P_t(c|I,x)$ for the body part labeling. These probabilities are combined to a global RDF probability $P(c|I,x)$ for the pixel (2). We take the most likely body part label, hereby assigning a single body part label to each pixel. If the probability distribution of the body part labeling does not produce a robust maximum likelihood estimate, the label which is most consistent with the pixel labeling from the eight neighboring pixels is chosen.

5. Body part proposals

At this point, the RDF has assigned part labels to each pixel in the image. Next, the set of pixel labels $C = \{c(x)\}$ must be clustered into a smaller set of part proposals V, from which a skeleton can be extracted. In this section, we describe the clustering process.

To efficiently cluster pixels, we utilize a breadth-first search (BFS) over a graph created by connecting neighboring pixels (remark again that the pixel also contains information on the depth and therefore directly corresponds to a voxel in 3D). Edges in the graph are removed if either the two nodes have different labels or if their L2 Euclidean distance between the pixels (voxels) is above a threshold. A filtering step removes all clusters that do not have a minimal number of nodes. This approach can be executed deterministically in $O(|V|+|E|)$ [41]. Additional stability is gained by performing this clustering in 3D rather than 2D. Once the first tree graph is searched, the segmentation continues with the next unprocessed point as a root node candidate. This process is iterated until all labeled pixels are processed.

A final filtering step calculates the statistics (centroid and covariance) of each cluster, and keeps those for which the first eigenvalue (largest dimension) is anthropometrically feasible for the body part length. A lower limit on the part length is not set, however, because there can be any number of reasons for which a part looks shorter than it should be (such as occlusions). Explicit occlusion reasoning could be added to improve this process.

6. Kinematic tree search

So far, the pixels in the image have been clustered and the clusters have been filtered to produce a large number of body part proposals. From these part proposals V we now generate proposals for all the people's locations and skeleton configurations S:

$$P(V|C,I) \;\rightarrow\; P(S|V) \tag{3}$$

This is the step in which each person is segmented separatly by extracting groups of clusters from the set of part proposals that conform to the kinematic constraints of the skeleton model, as in Fig. 5. This process is carried out in two phases with a dynamic programming approach.

Initially, the labeled labeled clusters are sorted into a 2D matrix based on size and label. The local consistencies between all pairs of part proposals are evaluated based on their 3D distances and covariances. Pairs of body parts are defined to be locally consistent if they have a parent–child (one step, $(n,n-1)$) or grandparent-grandchild (two-step, $(n,n-2)$) relationship in the kinematic model, and if their normalized distances $d_n(x_n,x_{n-1|2})$ are consistent with these relationships. The normalized distance $d_n(x_n,x_{n-1}) = d(x_n,x_{n-1}) - d_m(x_n,x_{n-1})$ is the difference between the distance between to parts d subtracted with the model distance (d_m) for this parent–child/grandchild. Allowing two degrees of separation (grandparent-grandchild) between parts increases robustness to missing parts due to noise or occlusion. Each parent node in the tree accumulates the likelihoods for each locally consistent link. This approach is similar to the Image Foresting Transform [42].

Next, globally consistent skeletons are extracted from the locally consistent pairs. A skeleton tree candidate is always rooted in either the two lower face parts or the neck, as these were empirically found to be the most stable. Skeleton tree candidates S_i are then accepted based on three evaluation parameters. The first evaluation parameter is the global error $\epsilon_g = \sum_p |d_n(p)|$ which is the sum of normalized distances for all parts p in the tree candidate. The second evaluation the normalized error $\epsilon_n = \frac{1}{n_p}\epsilon_g$ which is the global error on the tree candidate divided by the number of found parts n_p. The third evaluation parameter is the number of found body parts n_p that are consistent within that tree candidate. These three parameters are compared to a discrete threshold in order to accept a tree candidate all parameters must pass.

The output of this process is both the detection, segmentation, and the full poses of all the people in the image. If implemented with background subtraction only a single tree candidate will exist and each part will only occur once, simplifying ϵ_g. This, however, would be highly dependent on the quality of the background subtraction.

7. Appearance model estimation for segmentation refinement

After the first iteration of the pixel-wise body part labeling step (Section 4), the body-part proposal step (Section 5) and and the kinematic tree search step (Section 6), the obtained person detections and body part estimates are noisy and body parts are often missing, as shown in Figs. 2 and 6 in the left column.

On the other hand, false positive part detections rarely occur. Therefore this initial labeling (left column Fig. 6) can be used to estimate an appearance model of the human. One of the main contributions of our approach is the combination of RGB and depth information in online appearance model estimation to improve skeleton estimation and segmentation (see Fig. 7).

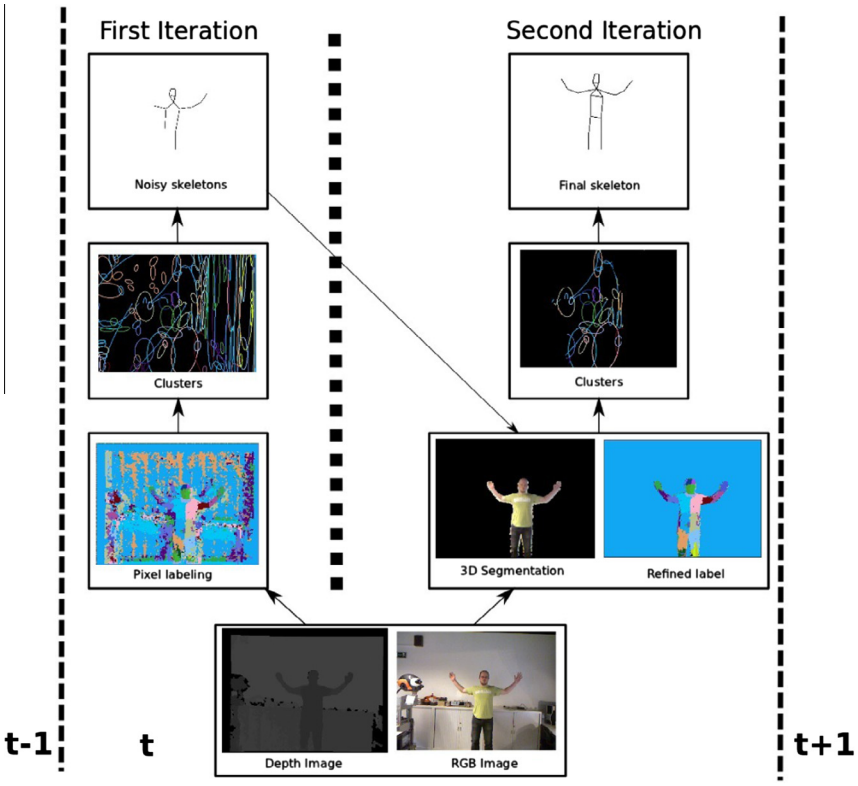

Fig. 6. Run-time system flow diagram. Two iterations take place during a single frame.

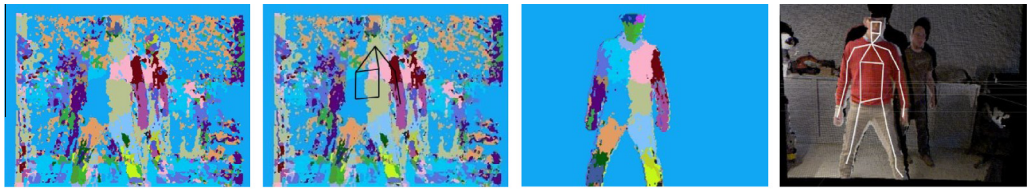

Fig. 7. Refinement through online appearance model estimation. (a) The original noisy labeling bootstraps online estimation of a color and depth-based appearance model, based on the first skeleton estimate (b), leading to the much cleaner labeling in (c) (shown for the front-most person). (d) shows the image as a slightly-rotated point cloud and the resulting skeleton of the front person.

From the initial segmentation of a person, a model is estimated of both bodypart locations in space and their appearance, given by the hue in the HSV space. This estimation is executed in a single frame iteration. In our current implementation the appearance model is not propagated over frames, although this could be a future improvement (tracking). Recently, Gulshan et al. [22] showed that a person could be segmented using GrabCut given a loose bounding box prior and a number of seed points. However, their approach did not use depth and GrabCut took six seconds to run, which is unacceptable for real-time tracking. Instead, we seed an Euclidean clustering based on the initial pixels' locations and hues. This can be made extremely efficient and effectively fills in missing pixels (Figs. 2 and 7). The features used for segmentation can be expanded as required by the application, for example by adding texture.

This new set of segmented pixels is then fed back into the part estimation and skeleton extraction process to produce a more robust estimate during a second iteration, as explained in the system overview (Section 3). By estimating an appearance model online, our algorithm exceeds the performance predicted by its training: it can detect poses that were not part of the training (see Section 8).

8. Results

Since we are not aware of data sets containing an annotated ground truth (like Vicon MOCAP data in combination with RGB-D images), the results presented in this paper are limited to the evaluation of the reliability in person and body part detection.

To test our algorithm, we collected 161 movies at 30 fps in indoor environments from a moving Kinect camera held approximately 1.4–1.8 meters above the ground. The 161 movies concern 38 different persons with about the same number of males and females, a wide range of ages (from approximately 22 to 50) and BMIs. These movies are a mixture of cluttered and uncluttered environments. Furthermore people's poses were not restricted, leading to a multitude of poses that were not in the training data. This scenario matches what would be seen from an indoor robotic platform such as the PR2 [43,44].

8.1. Comparison study

Our first evaluation is a comparison of our algorithm with the OpenNI NITE people tracking library [31]. Unfortunately, we were

unable to obtain an implementation of the Shotton et al. algorithm or their test data for comparison, nor were we able to obtain the source code for the NITE library.

The comparison is only done for a small sample set of 20 images as the comparison needs a manual overlay process. A representative sample image from the comparison versus NITE can be seen in Fig. 8(a), where our output and the output from NITE is overlayed on the original image. The results of our algorithm are shown as the white skeleton. The results of the NITE system are shown as a set of coordinate axis with one axis per joint. From the joint positions, we can see that our shoulder estimates and hip estimates are more accurate. In addition, with the NITE library, the orientation of the head is lost, while our system can recover the head orientation. Most importantly, the NITE system requires the user to stand in a Y-pose for tracking initialization, whereas our system detects and initializes tracking automatically.

To further compare our algorithm with existing algorithms we have looked for datasets to which existing algorithms have been

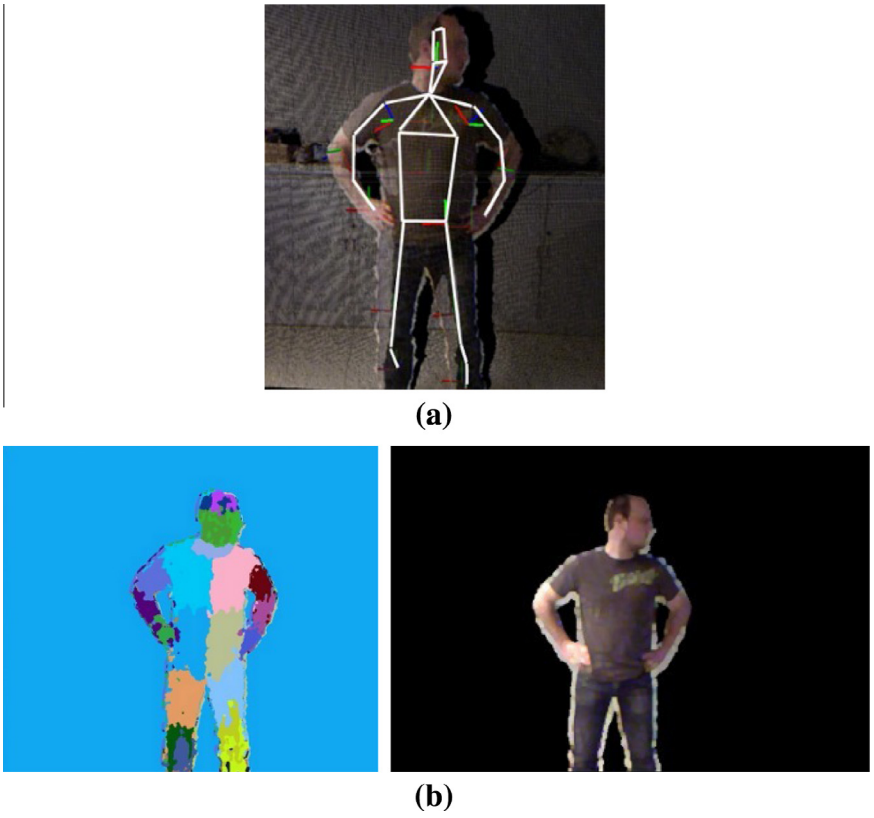

Fig. 8. (a) Comparison of the results of the NITE system and the proposed algorithm. The underlying image is an RGB-D frame shown in 3D and slightly rotated, leaving black shadows at depth discontinuities. The NITE results are shown as RGB axes for each joint. The results of our algorithm are shown as a white skeleton. Notice that our results show the head orientation, and provide a more accurate shoulder and hip placement. (b) Final body part labeling and final segmentation by our algorithm.

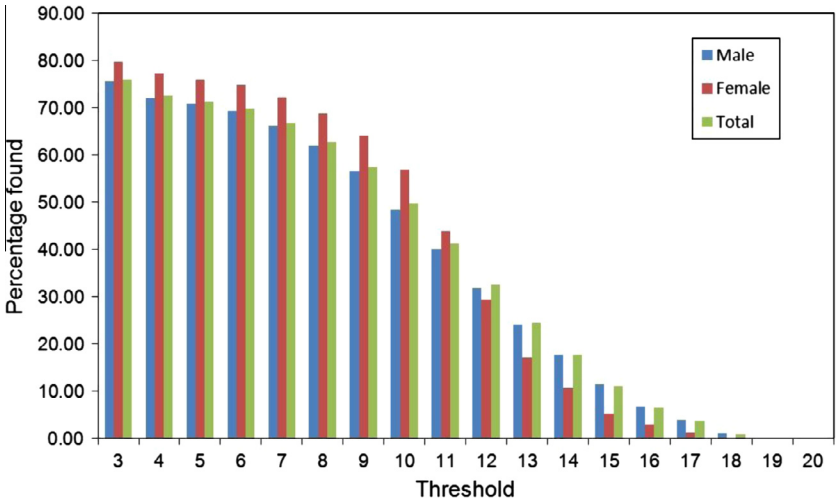

Fig. 9. Person detection rate versus the number of body parts threshold. For a skeleton to be declared a detection, a minimum number of body parts (out of 25) must be found. Increasing the threshold corresponds to requiring that more body parts of the person are detected.

applied. The data from [23] is the closest dataset aiming at the same objectives, however this data does not come from an RGB-D device but from a Swissranger TOF camera. Since the data is lacking RGB information and is of a much lower resolution as the data coming from the Primesense sensor in the Kinect, it is not suited for a fair comparison.

8.2. Person detection rate

As a second evaluation we study the person detection rate, defined as the percentage of images in which a person is detected out of all images containing a person. Since our algorithm did not produce any false positives (person detected when non was present) the person detection rate is equal to the recall for person detection (precision = 1). This lack of false positives is due to the selectiveness of the algorithm concerning: expected body part sizes, body part connections (two body parts can only be neigbors if this makes physically sense), and the kinematic model (the connections between all body parts) and the conservative threshold for the detected number of body parts in this kinematic model before accepting it as a person detection.

A set of 38 people in three different locations were (randomly) taken from the 161 movies. The people adopted different poses, producing 15,666 frames in both cluttered (as in Fig. 1) and and

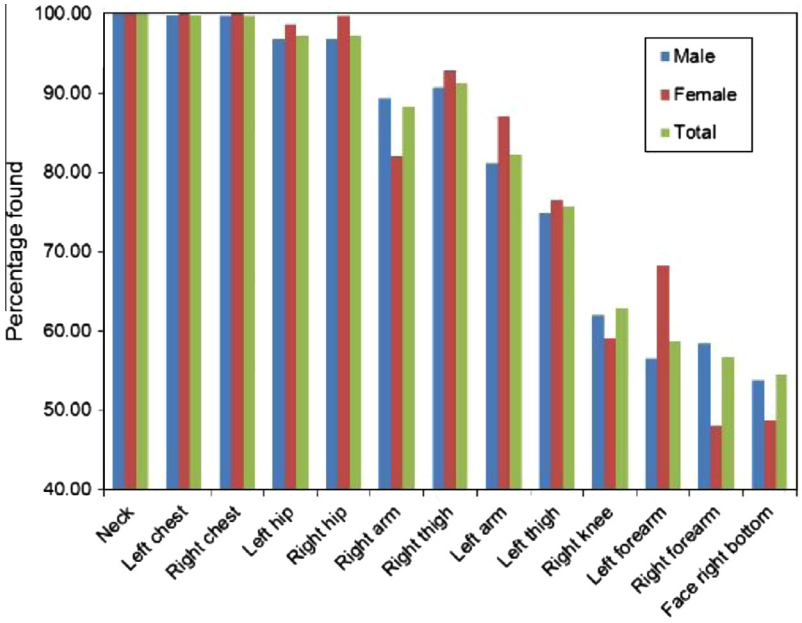

Fig. 10. Body part detection rate by body part with the number of body parts threshold set to eight. The neck and torso are reliable and anchor the skeleton. The arms and legs are less reliable, since they are often occluded or are in more complex poses. Although the head as a whole is extremely reliable, one of the four head parts, such as the 'face right bottom' part shown here, may be missing.

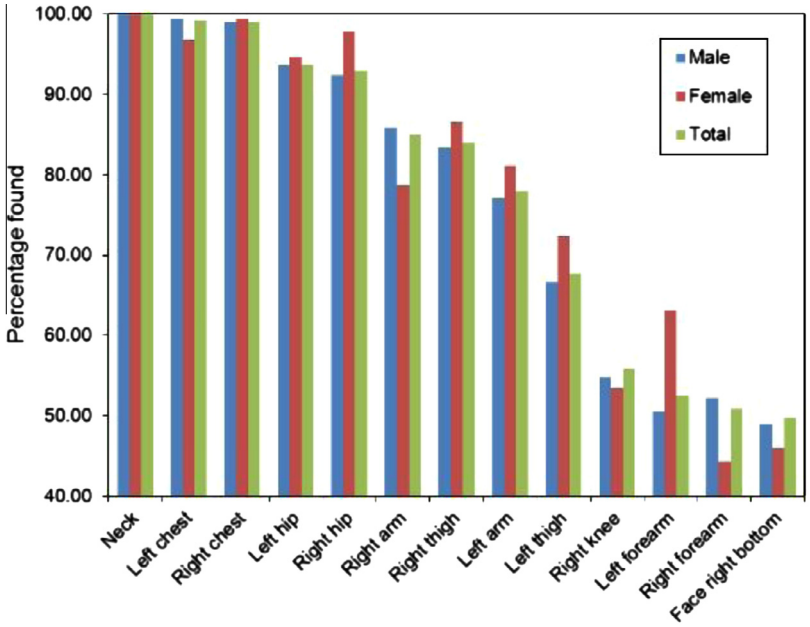

Fig. 11. Body part detection rate by body part with the number of body parts threshold set to four. As can be expected, the reliability of individual parts goes down as compared to the 8-part threshold case.

uncluttered environments. A skeleton (kinematic chain) was declared to be a person if it contained more than a threshold number of body parts. Skeletons that contained fewer parts were considered false positives and discarded. With this part threshold set to a default of four, the person detection rate was 73.95%. This is considered high due to the challenging nature of the dataset (clutter, occlusions, and persons partially in field of view).

Fig. 9 shows the effect of varying this number of body parts threshold. If more body parts are required to be visible, a smaller number of skeletons will be counted, leading to a lower acceptance rate. Remark that the number of visible body parts is also affected by the amount of occlusion.

Due to the support of the kinematic chain and on-line learning, the algorithm can handle subjects and poses that where not explicitly trained for. Fig. 12 shows the result for a small young female subject even though the camera angle for the image is very different from the straight-on, chest-height angle generated in the training set. The subject is segmented from the background and annotated correctly in a cluttered environment.

All the examples shown in this paper were obtained with the RDF trained for a single 25 year old slim male MakeHuman model in an uncluttered environment.

8.3. Body part detection rate for full body and upperbody

With the default number of body parts threshold of four, the recognition rate of individual parts on detected people can be seen in Fig. 10. Fig. 11 shows the same information for a threshold of 8 parts. We see here that the neck is the most stable feature, followed by the chest. This coincides with evidence from other pedestrian trackers, which tend to learn the head, neck and shoulder area, and reduce focus on the rest of the body [45]. Comparing the two graphs also illustrates that when the threshold is raised,

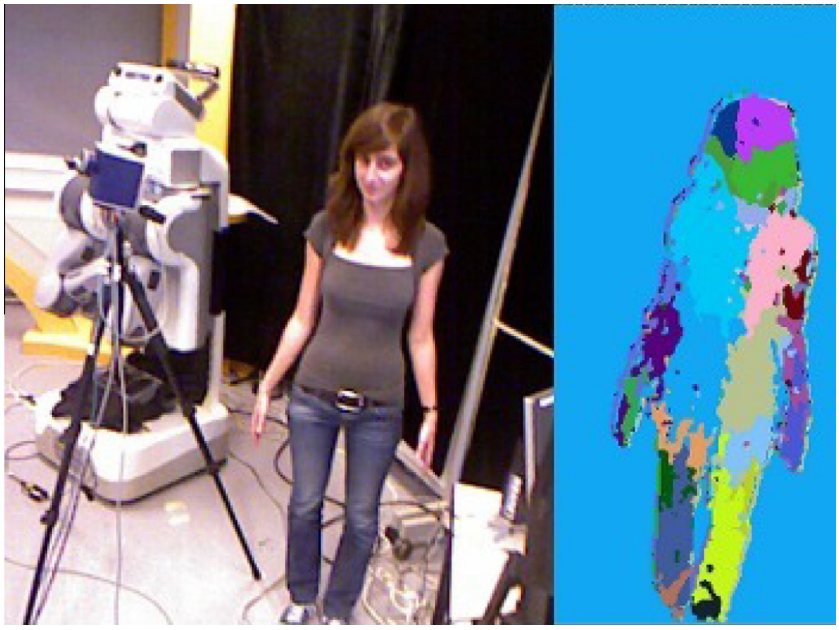

Fig. 12. Body part labeling results of a young small female subject in a cluttered environment. The camera angle for this image is very different from the straight-on, chest-height angle generated in the training set. The training set only contained images of a 25 year old slim male MakeHuman model in an uncluttered environment.

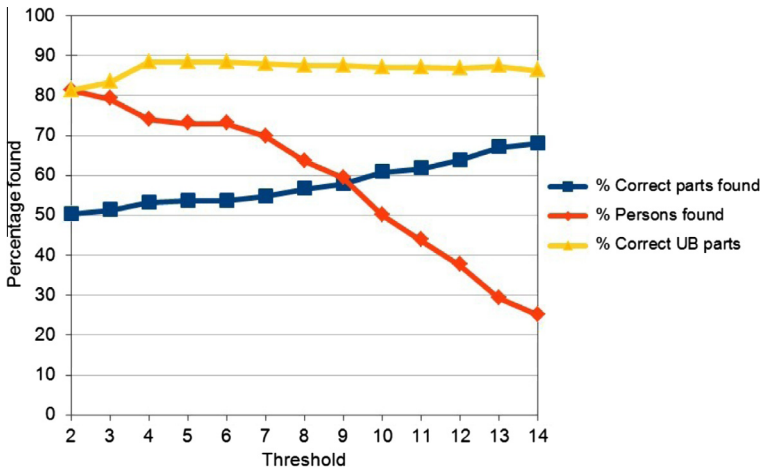

Fig. 13. The person detection rates (red, diamonds) and the body part detection rates for the full body (blue, squares) and the upperbody (yellow, triangles) as a function of the body part threshold for the 96 random frames, for "the all poses set". (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the detection in general becomes more robust (fewer false positives for the body parts).

As a third evaluation we study the body part detection rate for the body parts of the full body and for the body parts of the upperbody. The body parts of the upperbody are explicitly evaluated because of their importance for personal robotics applications. The body part detection rate is defined as the number of detected body parts out of the number of visible body parts. This reflects the recall for the body part detection since our algorithm did not produce any false positives (precision = 1).

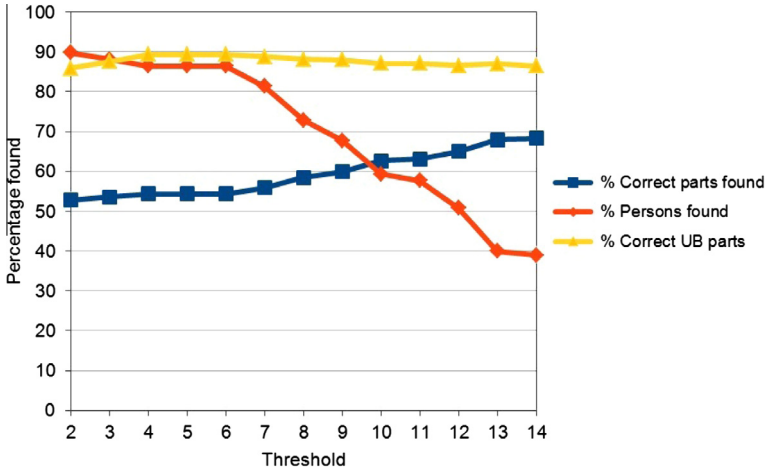

Fig. 14. The person detection rates (red, diamonds) and the body part detection rates for the full body (blue, squares) and the upperbody (yellow, triangles) as a function of the body part threshold for the 96 random frames, for "the trained poses set". (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

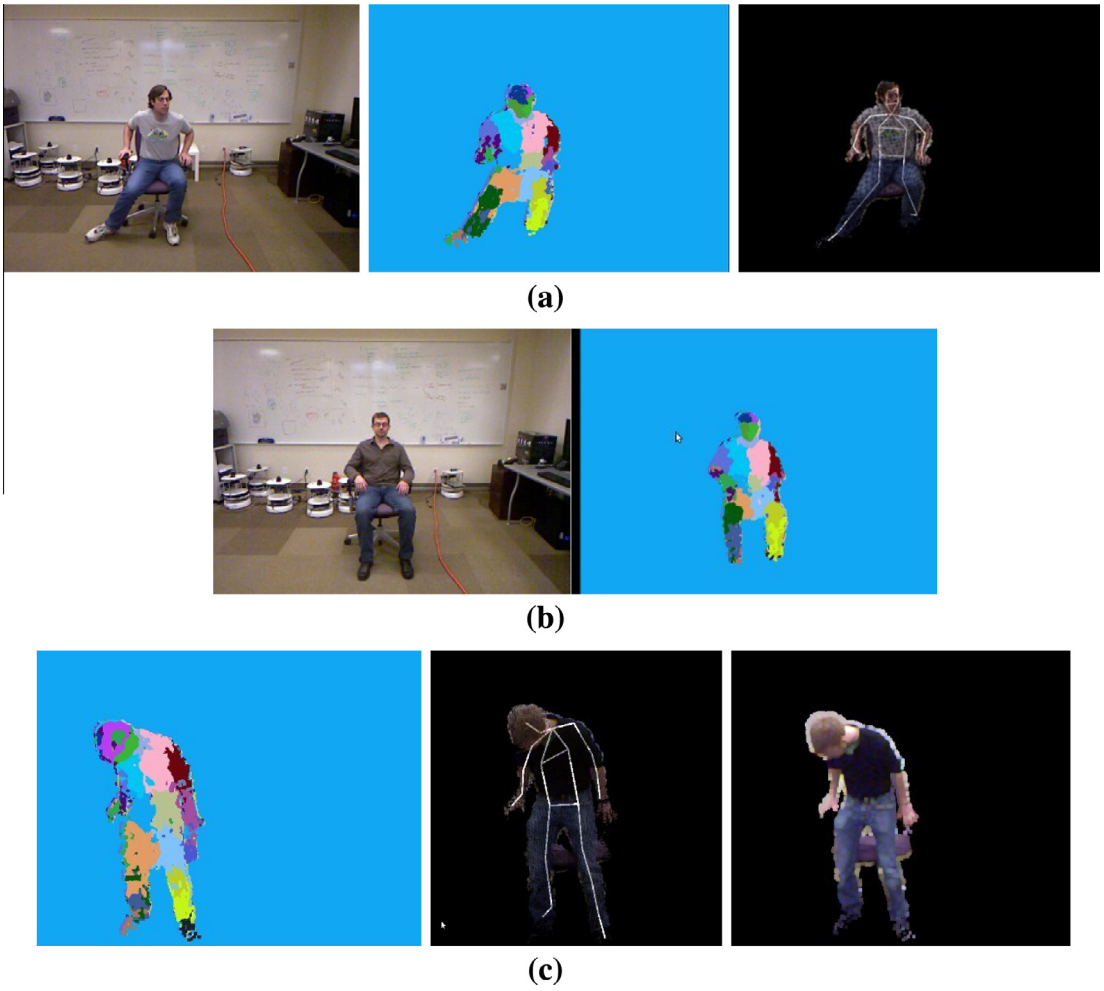

Fig. 15. Results for poses that were not in the training set, but were still tracked adequately. The top row shows two sitting poses which are made more difficult by the presence of the chair. The bottom row shows someone who is going to sit down.

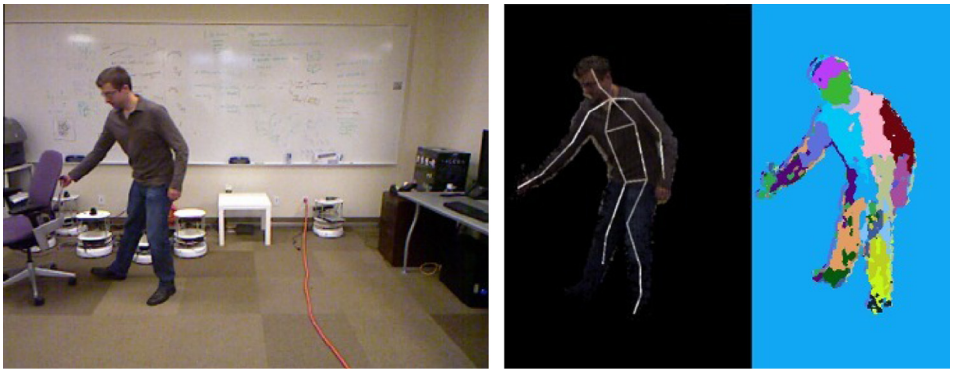

Fig. 16. Result for a pose that was similar to a pose in the training dataset. (a) The original image. (b) The skeleton overlayed on the person segmented from the background, and the final body part labeling.

After this global automated evaluation, a random subset of 100 sample frames were taken out of the 15666 frames (as defined in the previous section (Section 8.2)). These frames contain both poses present in the training data as well as untrained poses. We refer to this set as the "all poses set". These frames were visually verified and annotated. The annotation included: a count of the body parts correctly found, a count of the visible body parts, a count of the visible upperbody parts (these were defined as the 17 parts above the pelvis), and whether the pose was close to one in the training dataset. During this annotation, four frames were found that did not have a person visible in them, these frames were excluded. Remark that all the frames in the dataset that did not have people in them (including those with clutter) were correctly labeled as empty.

Fig. 17. Tracking a person in a highly cluttered environment with a high degree of occlusion.

Fig. 13 shows the person detection rates (red, diamonds) and the body part detection rates for the full body (blue, squares) and the upperbody (yellow, triangles) as a function of the number of body part threshold for the 96 random frames. The body part detection rate for the upperbody stays fairly constant between 80% and 90%, while the body part detection rate for the full body is considerably lower. This illustrates that most of the problems occur in the lower body parts. This can be partially explained by the influence of the ground plane on the labeling, as well as by occlusions, which mainly occur at lower body part level in indoor environments. As the algorithm does not use background/foreground segmentation of the person, pixels from the ground plane and occlusions are also processed. The training data (Section 4.1) however does not contain a ground plane or occlusions. Therefore, the ground plane and occlusions have a significant negative effect on the labeling.

Fig. 14 shows the same statistics as Fig. 13 but now for 59 poses that closely resembled a pose in the training data. We refer to this set as the "trained poses set". The person detection rate drops less when increasing the threshold compared to the "all poses set". Furthermore the body part detection rate of the upperbody is also slightly higher. In these "trained poses set" no occlusions were present, however the biggest influence on wrong pixel labeling was the ground plane (also visible in Fig. 18). This can be explained

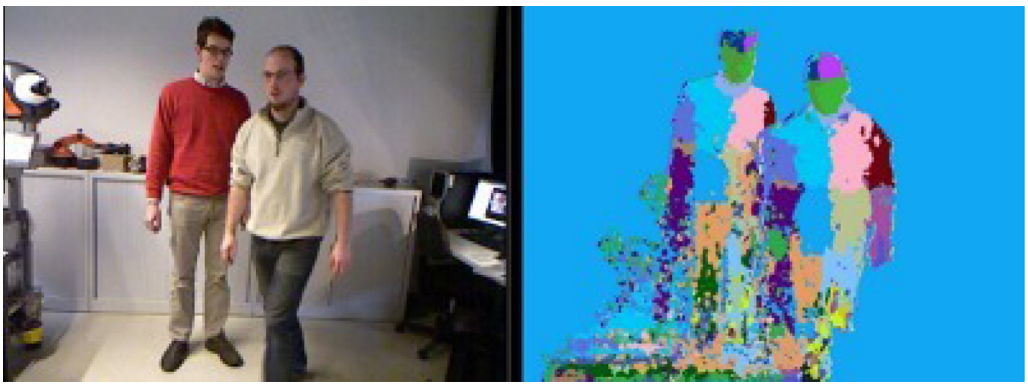

Fig. 18. The system can successfully detect multiple people in close proximity. The similar color of the person's pants and the cabinet and floor, and the proximity of the other person confuses the pixel labeling. This confused pixel labeling however does not lead to false positives in the final skeletons.

because of the lack of ground plane in the training data. In future work we will investigate if the ground plane can be included in the training data.

The 10 percent of non detected upperbody parts in both the "all poses" and "trained poses" set are due to the small size of the hands and the elbows. The hands are often not correctly detected because of the diversity in poses they can adopt, some of which make them look as a rigid extension of the lower arm (like making a fist) in the depth image. This could be improved when a better estimate of the lower arm length is available. The elbows are often ignored because the algorithm includes a heuristic rule that if the uncertainty of the elbow region is too high, it will just ignore it and connect the upper arm to the lower arm. This could be improved by explicitly taking into account the limited number of pixels covering the elbow.

In all of the frames in both the "all poses" and "trained poses" set *no false positives* were detected, which indicates that our algorithm has a high reliability on the parts found (precision = 1).

8.4. Visual evaluation

For the third evaluation, subjects were asked to perform 3 motions in front of the camera: (1) grabbing something out of their pocket and handing it to the camera, (2) walking around and (3) sitting down on a chair. Examples of these tests and results can be seen in Figs. 15(a–c), 16 and 17. Since that the training motion capture data did not include people sitting nor any chairs, this test data provided two challenges which our system successfully dealt with.

In future work we will investigate whether the performance can be further improved by adding objects into the training data [46] depending on the application.

Finally, we performed a premature qualitative multi-person evaluation. In this test subjects walked around each other, creating inter-occlusions. This required the system to find multiple skeletons among a larger body part set, as in Fig. 18. The thorough qualitative and quantitative evaluation of the proposed algorithm for multi-person detection will be subject of future work.

8.5. Speed

This complete runtime process (body part proposals, kinematic tree search and appearance modeling) happens efficiently in a CUDA GPU implementation at 17–20 fps on a regular CPU-GPU combination (on a dual core CPU with a GTX550ti Nvidia GPU) and is considered real-time for personal robotics applications.

9. Conclusions

In this paper, we have presented a system for person detection, segmentation, and skeleton tracking for RGB-D data that is opportune for both indoor mobile robots and many other applications. The power of the proposed approach is its customizability and adaptability to different applications and environments through lowered training requirements, modularity, and an open source codebase. In addition, background subtraction is not required, enabling work on moving platforms and among clutter. By on-line estimation of a model of combined appearance and depth, our system can surpass its training data and adapt to new people and environments.

Our code, data, training and documentation will be open and freely available.

As future work, the authors will validate the developed system quantitatively using ground truth marker data (for instance obtained using a Vicon system) and using a recently published RGB-D markerless dataset, containing Kinect images of humans in annotated poses.

Furthermore, we will investigate if incorporating occlusion reasoning even lead to greater environment adaptability. Finally, we will collect training data for specific applications and train a specific RDF for these applications.

We hope that our open system will encourage community participation and improvements.

For additional videos, results and details, please visit http://people.mech.kuleuven.be/kbuys/jvci/ and http://www.pointclouds.org.

Acknowledgments

The authors would like to acknowledge Willow Garage for their open contribution to the community with the PR2 Beta Program, Nvidia for their financial contributions and technical support for this project. The Flemish and the German government for financially supporting the authors.

Koen Buys is funded by KU Leuven's Concerted Research Action GOA/2010/011 Global real-time optimal control of autonomous robots and mechatronic systems, a PCL-Nvidia Code Sprint grant, an Amazon Web Services education and research grant, this work was partially performed during an intern stay at Willow Garage.

Anatoly Basheev is funded by Nvidia as support for the PointCloud Library.

Tinne De Laet is a PostDoctoral Fellow of the Research Foundation Flanders (FWO) in Belgium.

The authors would like to acknowledge their interns, Inge Famaey, and Jorn Wijckmans for their assistance in the data collection. Finally we thank all the participants for their cooperation.

References

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, Real-time human pose recognition in parts from a single depth image, in: CVPR, 2011.
[2] Microsoft XBOX Kinect, <xbox.com>, 2010.
[3] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, A. Fitzgibbon, Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera, in: UIST, 2011.
[4] V. Lepetit, P. Lagger, P. Fua, Randomized trees for real-time keypoint recognition, in: CVPR, 2005.
[5] J. Shotton, M. Johnson, R. Cipolla, Semantic texton forests for image categorization and segmentation, in: CVPR, 2008.
[6] T. Sharp, Implementing decision trees and forests on a gpu, in: ECCV, 2008.
[7] J.R. Quinlan, Induction of decision trees, Machine Learning, vol. 1, Kluwer Academic Publishers, 1986, pp. 81–106.
[8] T. Moeslund, A. Hilton, V. Kruger, A survey of computer vision-based human motion capture, CVIU 2001, Elsevier.
[9] T. Moeslund, A. Hilton, V. Kruger, A survey of advances in vision-based human motion capture and analysis, CVIU 2006, Elsevier.
[10] R. Poppe, Vision-based human motion analysis: an overview, CVIU 2007, Elsevier.
[11] S. Ioffe, D. Forsyth, Probabilistic methods for finding people, IJCV 43 2001, Springer.
[12] G. Mori, J. Malik, Estimating human body congurations using shape context matching, in: ICCV, 2003.
[13] D. Ramanan, D. Forsyth, Finding and tracking people from the bottom up, in: CVPR, 2003.
[14] A. Agarwal, B. Triggs, 3d human pose from silhouettes by relevance vector regression, in: CVPR, 2004.
[15] L. Sigal, S. Bhatia, S. Roth, M. Black, M. Isard, Tracking looselimbed people, in: CVPR, 2004.
[16] P. Felzenszwalb, D. Huttenlocher, Pictorial structures for object recognition, IJCV 2005, Springer.
[17] R. Urtasun, T. Darrell, Local probabilistic regression for activity independent human pose inference, in: CVPR, 2008.
[18] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, P. Torr, Randomized trees for human pose detection, in: CVPR, 2008.
[19] Z. Tu, Auto-context and its application to high-level vision tasks, in: CVPR, 2008.
[20] L. Bourdev, J. Malik, Poselets: body part detectors trained using 3d human pose annotations, in: ICCV, 2009.
[21] Y. Zhu, K. Fujimura, Constrained optimization for human pose estimation from depth sequences, in: ACCV, 2007.

[22] V. Gulshan, V. Lempitsky, A. Zisserman, Humanising grabcut: learning to segment humans using the kinect, in: Consumer Depth Cameras for Computer Vision at ICCV, 2011.
[23] V. Ganapathi, C. Plagemann, D. Koller, S. Thrun, Real-time motion capture using a single time-of-flight camera, in: CVPR, 2010.
[24] C. Plagemann, V. Ganapathi, D. Koller, S. Thrun, Real-time identification and localization of body parts from depth images, in: ICRA, 2010.
[25] M. Siddiqui, G. Medioni, Human pose estimation from a single view point, real-time range sensor, in: CVCG at CVPR, 2010.
[26] E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3d mesh segmentation and labeling, ACM Trans. Graphics 29 (3) (2010).
[27] M. Ye, X. Wang, R. Yang, L. Ren, M. Pollefeys, Accuracte 3d pose estimation from a single depth image, in: ICCV, 2011.
[28] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, A. Fitzgibbon, Efficient regression of general-activity human poses from depth images, in: ICCV, 2011.
[29] H. Ning, W. Xu, Y. Gong, T.S. Huang, Discriminative learning of visual words for 3d human pose estimation, in: CVPR, 2008.
[30] R. Okada, S. Soatto, Relevant feature selection for human pose estimation and localization in cluttered images, in: ECCV, 2008.
[31] Primesense, <primesense.com>, 2010.
[32] M. Bastioni, M. Flerackers, J. Capco, MakeHuman, <makehuman.org>, 2012.
[33] D.V. Deun, V. Verhaert, K. Buys, B. Haex, J.V. Sloten, Automatic generation of personalized human models based on body measurements, in: International Symposium on Digital Human Modeling, 2011.
[34] K. Buys, D.V. Deun, T.D. Laet, H. Bruyninckx, On-line generation of customized human models based on camera measurements, in: International Symposium on Digital Human Modeling, 2011.
[35] A. Weiss, D. Hirshberg, M. Black, Home 3d body scans from noisy image and range data, in: ICCV, 2011.
[36] CMU Graphics Lab Motion Capture Database, <mocap.cs.cmu.edu/>, 2011.
[37] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, in: OSDI, 2004.
[38] Hadoop, <hadoop.apache.org>, 2011.
[39] D. Borthakur, The hadoop distributed file system: architecture and design, Tech. rep., <Apache.org>, 2007.
[40] T. White, third ed., Hadoop: The Definitive Guide, O'Reilly, 2012.
[41] D. Merrill, M. Garland, A. Grimshaw, High performance and scalable gpu graph traversal, Technical Report CS-2011-05, University of Virginia, Augustus 2011.
[42] A. Falcao, A. Falc, et al., The image foresting transformation, IEEE Transactions on Pattern Analysis and Machine Intelligence, Citeseer, 2000.
[43] K. Wyrobek, E. Berger, H.M.V. der Loos, J.K.S. Jr., Towards a personal robotics development platform: rationale and design of an intrinsically safe personal robot, in: Proceedings of the International Conference on Robotics and Automation (ICRA), 2008.
[44] A. Oyama, K. Konolige, S. Cousins, S. Chitta, K. Conley, G. Bradski, Come on in, our community is wide open for robotics research!, in: RSJ, <http://www.rsj.or.jp/rsj2009/>, 2009.
[45] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 2005.
[46] J. Stckler, N. Biresev, S. Behnke1, Semantic mapping using object-class segmentation of rgb-d images, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.