Neurocomputing 147 (2015) 174–184

Contents lists available at ScienceDirect

Neurocomputing

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

Self-organizing maps for hand and full body tracking

Foti Coleca [a,c,*], Andreea State [a,b], Sascha Klement [c], Erhardt Barth [a], Thomas Martinetz [a]

[a] Institute for Neuro- and Bioinformatics, University of Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany[1]
[b] University "POLITEHNICA" of Bucureşti, Splaiul Independenţei 313, 060042 Bucureşti, Romania[2]
[c] gestigon GmbH, Maria-Goeppert Straße 9a, 23562 Lübeck, Germany[3]

ARTICLE INFO

Article history:
Received 10 April 2013
Received in revised form
27 October 2013
Accepted 31 October 2013
Available online 20 June 2014

Keywords:
Body tracking
Hand skeleton tracking
Gestures
Self-organizing maps
Kinect
TOF cameras

ABSTRACT

Touch-free gesture technology opens new avenues for human–machine interaction. We show how self-organizing maps (SOM) can be used for hand and full body tracking. We use a range camera for data acquisition and apply a SOM-learning process for each frame in order to capture the pose. In a next step we introduce an extension of the SOM to 1D and 2D segments for an improved representation and skeleton tracking of body and hand. The proposed SOM based algorithms are very efficient and robust, and produce good tracking results. Their efficiency allows to implement these algorithms on embedded systems, which we demonstrate on an ARM-based embedded platform.

1. Introduction

The challenge of human hand/body tracking and pose estimation has gained much attention during the last years, mainly driven by the mainstream interest toward building usable gestural interfaces for consumer applications. This was seen with the introduction of gaming consoles that can track a user's hand gestures (Nintendo Wii) or body (Microsoft Kinect), which showed that gesture interfaces can be used to create rich interactive experiences. Hand tracking alone can be used in a wide variety of applications and represents a milestone in human–machine interaction. A major catalyst was the introduction of new technologies and devices designed for 3D image acquisition. Depth cameras provide a more favorable framework for tracking algorithms, simplifying the task of three-dimensional model fitting, giving algorithms that use them an edge over 2D image processing techniques.

Nevertheless, these are both difficult problems, especially estimating the hand pose: the hand itself is a complex object, having an extremely large state space due to its 27 degrees of freedom [1]. Because of this complexity, its projection in images often involves self-occlusions which, coupled with the chromatic uniformity of the skin, makes segmentation and feature detection very difficult. With speeds reaching up to 5 ms^{-1} for translation and 300°s^{-1} for wrist rotation [2], consecutive frames of a moving hand can have very little in common (especially with a slow camera frame rate), making it a difficult object to track. Adding to these difficulties, the algorithms have to cope with various backgrounds and lighting conditions.

3D cameras can alleviate some of the difficulties described above, having multiple advantages over standard color image processing. A critical step of any pose estimation algorithm is object segmentation. By having access to the depth map of the scene, objects can be segmented accurately based on their shape and distance to the camera, regardless of texture, skin color or background clutter. With active 3D technologies (such as time-of-flight or structured light), there is even no need for a uniform or consistent scene illumination. This is a very useful feature for real-world applications, where consumer devices are being used by a variety of people in a variety of environments. Our work focuses on building a hand/body pose estimation and tracking algorithm for such a 3D camera, that is both accurate and has low computational costs.

In this article we present an extension of the hand pose estimation method proposed in [3], as well as a practical implementation of the algorithm. It is all based on the original work introduced in [4], a novel approach to pose estimation by the use of self-organizing maps (SOM) [5] to fit a topology of the human

* Corresponding author at: Institute for Neuro- and Bioinformatics, University of Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany.
E-mail addresses: coleca@inb.uni-luebeck.de (F. Coleca), state@inb.uni-luebeck.de (A. State), sascha.klement@gestigon.de (S. Klement), barth@inb.uni-luebeck.de (E. Barth), martinetz@inb.uni-luebeck.de (T. Martinetz).
[1] http://www.inb.uni-luebeck.de.
[2] http://www.upb.ro.
[3] http://www.gestigon.de.

http://dx.doi.org/10.1016/j.neucom.2013.10.041
0925-2312/

upper body inside a 3D point cloud. We show that this topology can be successfully extended to a full body as well as a human hand. A further extension of the algorithm is presented, in which the original SOM is extended to include not only nodes but also the segments and planes between the nodes of the topology. This has the advantage of requiring less nodes than the original topology, offering a more realistic representation of the human hand and being more stable overall.

The algorithms are of a low enough computational cost that they can be implemented on an embedded platform and used to track subjects in real time. We will show an implementation of both the hand/body SOM and the segment-plane extension for the hand on an OMAP-4430 powered Pandaboard, using time-of-flight (PMD camboard) or structured light (Microsoft Kinect) cameras as an input device for 3D data. Our algorithm is able to track the user at the native framerate of the camera.

2. Related work

The most commonly used methods to gather accurate data for skeleton tracking are marker-based motion capture systems in the case of the whole-body skeleton or by the use of a "data glove" for hand pose estimation [6]. These methods are cumbersome and can be used only in controlled environments. Thus, marker-less pose estimation is a heavily researched area in image processing — recent surveys cite dozens of papers [2] on hand pose estimation and several hundred [7] on human motion capture and analysis.

For example, the authors of [8] use kinematic models and build a hand state model, which consists of a set of lines and points generated by the projection of the hand model to the image plane. Hand pose estimation based on features derived from projections of the hand and its shadow is presented in [9]. The method requires controlled background and lighting and is susceptible to occlusion. In [10] and [11], the authors use a feature extraction approach based on Curvature Scale Space to achieve translation, scale and rotation invariant recognition of hand postures. Again, the method is tested in a controlled environment, as it requires an accurate segmentation of the hand contour.

The authors of [12] introduce a machine learning architecture for matching image features to 3D hand example poses, which requires to solve an optimization problem based on Bayes' rule. Another approach is to estimate the hand pose with a database of synthetic hand images. For instance, in [13] an indexed image database is used to retrieve the closest hand match, with an adapted chamfer distance and line matching algorithm. In [14], the authors implement a cascade of increasingly complex classifiers to determine the hand pose from synthetic training data. In order to better handle occlusions, particle filters can be used. In [15], the authors apply a meta-descent algorithm to minimize the distance between a predicted position and the observed position, while particle filters predict new sample positions and help the optimization algorithm to recover from local minima. As shown in [16], the combined usage of intensity images and range information provides a good framework for body tracking.

Regarding performance, most algorithms surveyed by Erol et al. [2] stay below 30 frames per second (which we regard as being real-time), with only one exception [17]. Other solutions leverage the computing power of the GPU in order to achieve high frame rates [18–20]. Most existing approaches are aimed at high-performance desktop machines.

3. The SOM tracking algorithm

The node-based SOM tracking algorithm (which we will refer from now on as the Standard SOM Algorithm) starts with the initialization of its network weights, followed by the iteration of two steps: the competition and the update of the weights. At every iteration, a sample point from the dataset is randomly chosen. First, during the competition phase, a winner node (i.e. the weight with the minimum Euclidean distance to the sample point) is computed.

Given a network with n neurons and a sample point $\mathbf{x} \in \mathbb{R}^3$, we determine the winner node $\hat{i}$ as follows:

$$\hat{i} = \arg\left\{ \min_i \|\mathbf{x} - \mathbf{w}_i\|_2 \right\},\ i = 1, \ldots, n \tag{1}$$

with $\mathbf{w}_i \in \mathbb{R}^3$ being the weight of node i. Next, the update phase aims at decreasing the distance between the winner-node weight and the sample point, by an amount given by the learning rate $\epsilon(t)$. First, let us define the learning rate function as

$$\epsilon(t) = \epsilon_i \left(\frac{\epsilon_f}{\epsilon_i} \right)^{t/t_{max}}, \tag{2}$$

where ϵ_i is the initial learning rate, ϵ_f is the final learning rate, t is the current iteration, and t_{max} is the maximum number of iterations performed on the network. Then, the weight $\mathbf{w}_{\hat{i}}$ is updated at step t according to

$$\mathbf{w}_{\hat{i}}(t+1) = \mathbf{w}_{\hat{i}}(t) + \epsilon(t)(\mathbf{x} - \mathbf{w}_{\hat{i}}(t)). \tag{3}$$

The standard SOM algorithm then also applies a neighborhood update, in the sense that not only the winner-node weight is updated, but also the weights of the neighbor-nodes (in general with a smaller learning rate). In our case we updated only the direct topological neighbors of the winner node according to (3), but with a learning rate of $\tilde{\epsilon}(t) = \epsilon(t)/2$.

These steps are repeated for hundreds or thousands of iterations. This makes the skeleton graph fit to the point cloud and stay within its confines.

4. Topology expansion

First, we expand the 44-node upper body topology presented in [4] (Fig. 1a) to two topologies, one representing the whole body (Fig. 1b), and the other representing the human hand (Fig. 1c). The models were chosen so they mimic the anatomical landmarks of their real-world counterparts – limbs and joints for the body and phalanges and interphalangeal joints for the hand. The rigid bodies (torso and palm) are modeled as a mesh. Both produce good qualitative results in our implementation. The end results achieved with the standard SOM for the hand and body are shown in Figs. 2 and 3, showcasing the method's robustness.

It can be seen that the hand tracker is able to cope with missing data (Fig. 2b,c as white areas on the palm), the skeleton's topology remaining stable, the fingers being retracted in the palm. This is considered to be correct behavior, as the fingers will be reported as "bent" to a subsequent gesture recognition algorithm.

For the full-body tracker, the topology is robust enough to perform a good fitting over the subject's body, even when there is occlusion occurring. This is shown in Fig. 3b–d: it can be seen that when the user crosses his arms in front of him the skeleton retains its geometry afterwards, even if one of the arms occludes the other. This is true also for the rest of the topology nodes, such as the torso. Fig. 3a shows how the skeleton tracks the body shape in 3D, following the user's leg even though it is not in the same plane as the body.

Such configurations would be very hard to track using just a 2D image, particularly because of the juxtaposition of the hands and torso. This issue is resolved by using a 3D camera, which can differentiate between surfaces of various depths.

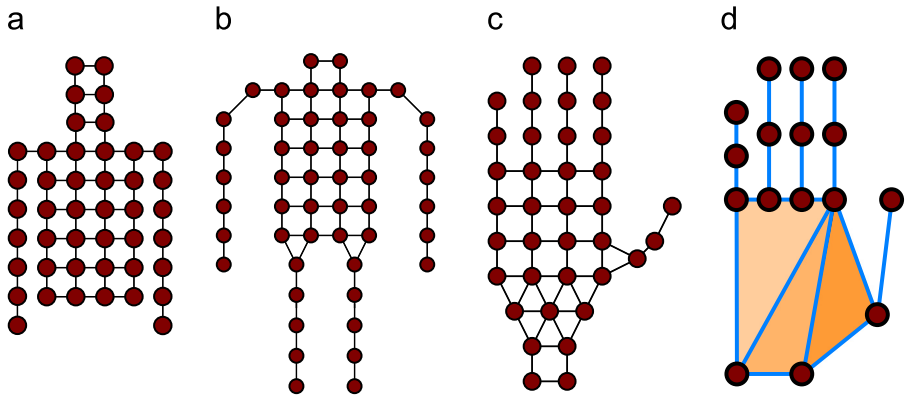

Fig. 1. SOM topologies: (a) The upper body topology proposed in [4]; (b) and (c) expanded topologies for the whole body and the hand; and (d) the proposed extended SOM hand topology for segments and planes.

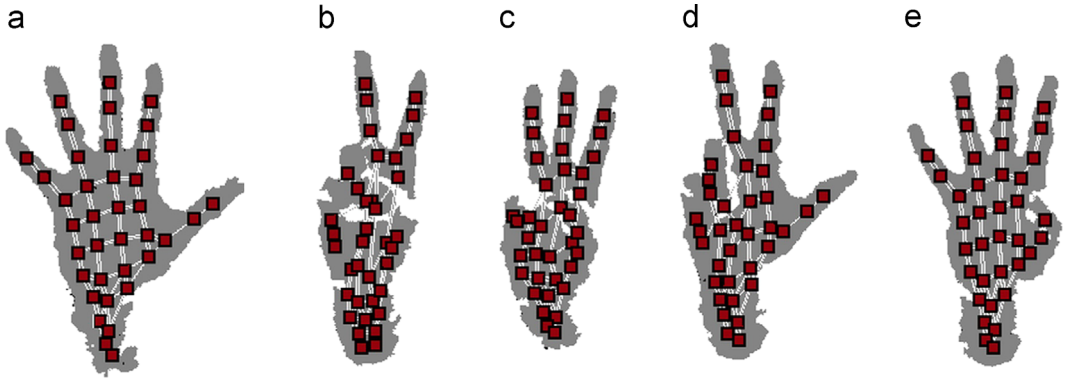

Fig. 2. The SOM skeleton results for various hand poses.

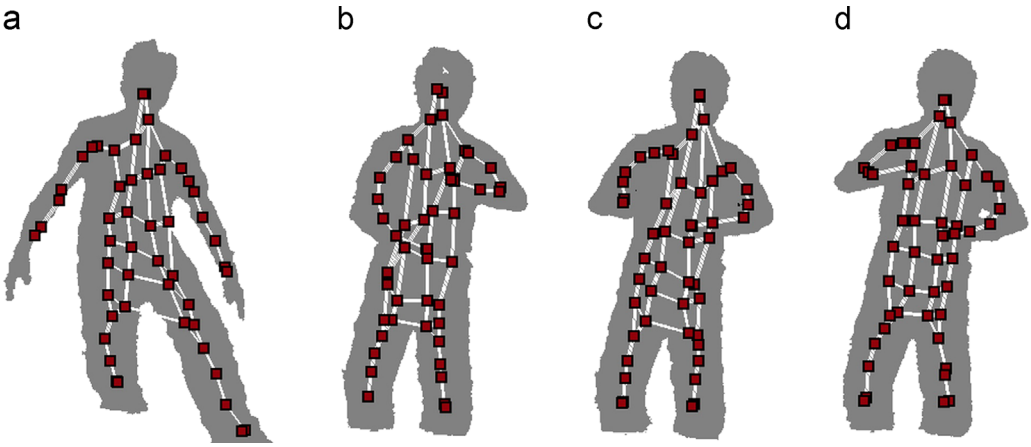

Fig. 3. The SOM skeleton results for various body poses.

This approach is robust enough for most poses, but does not guarantee that the topology will resemble the actual pose. We provide the example of a subject that brings his arms close to his body as to create a continuous region in 3D space, then extending them to his sides. Sometimes it happens that the extremities of the topology continue to be attracted by the large amount of points in the torso and subsequently remain in that area. This results in a topology following the arm's shape up to the last node, which, instead of being in the region of the subject's palm, will be inside his torso, connected by a very long edge.

Generally, the problem fixes itself after some time, but to speed recovery, we introduce an update rule that limits the length of the edges that connect the nodes, effectively limiting cases like the one described. At every update step the distance $d(\mathbf{w}_{\hat{i}}, \mathbf{w}_{\hat{i}a})$ between the current node $\mathbf{w}_{\hat{i}}$ and a specific neighbor $\mathbf{w}_{\hat{i}a}$, referred to as an "anchor", is calculated. If it exceeds a certain threshold θ, we set

$$\mathbf{w}_{\hat{i}} = \mathbf{w}_{\hat{i}a} + \theta \cdot \frac{(\mathbf{w}_{\hat{i}} - \mathbf{w}_{\hat{i}a})}{\|\mathbf{w}_{\hat{i}} - \mathbf{w}_{\hat{i}a}\|_2}. \tag{4}$$

This rule ensures a maximum distance θ between the current node $\mathbf{w}_{\hat{i}}$ and the anchor $\mathbf{w}_{\hat{i}a}$. This will prevent very large distances to occur between nodes. The anchor is selected for each node as

the one of the node's neighbors that is closest to the center of the topology.

Such a correction works well for most cases, but sometimes structural deficiencies of the topology of the standard SOM occur. Taking simple nodes (zero-dimensional objects) for representing extended three-dimensional objects has intrinsic deficits. When trying to fit the topology inside the point cloud, its behavior is not always consistent to that of the object it is tracking. Sometimes nodes which are responsible for the arm are closer to data points of the torso than torso nodes, thus resulting in an unstable topology. This is more apparent in the case of the hand, where the fingers are very close and can touch and occlude each other during normal hand movements.

In the next section, we propose a possible solution, modifying the topology so that not only the nodes will participate in the competition and update phase, but also the segments (one-dimensional objects) between the nodes and planes (two-dimensional objects) defined by them. This results in a much better representation of the object that we want to track and also provides us with the means to apply further corrections or constraints on an anatomical and anthropometric basis. We present this extension only for the human hand, as it is influenced to a greater degree by the problem described above, but it can be applied to the body skeleton as well.

5. The extended SOM

Our proposed algorithm extends the competition and the update step to 1D and 2D network segments. The 1D-segments are the lines between pairs of connected nodes, and the 2D-segments are the triangles determined by triples of connected nodes. 1D-segments allow for a more accurate representation of the fingers, and the 2D-segments model the palm of the hand. By segment updates we aim at minimizing the average distance between network segments and points from the dataset (point cloud of the hand provided by the 3D-camera). We now have not only elements of dimension zero (nodes) like in the standard case described in the last section, but also elements of dimension one and two for representing the data distribution. Since the new segments and planes approximate the data over a larger area, we can reduce the topology to a simpler one. The fingers will have two segments instead of three (the thumb one instead of two), and the mesh that modeled the palm can be replaced by three triangles. The new topology can be seen in Fig. 1d.

This approach is motivated by the fact that a hand-like topology involves a difficult separation between the nodes corresponding to different fingers. A node that belongs to one finger can easily be attracted by another finger, given the topological closeness. This may lead to an erroneous tracking of the hand and destroy the topological relations. With the 1D and 2D segments we can represent fingers and parts of the palm more accurately and expect the self-organizing maps to be less prone to this type of errors.

5.1. The extended SOM algorithm

The competition phase in our extended SOM algorithm determines whether a single node or a 1D-segment or a 2D-segment is closest to the randomly chosen sample point $\mathbf{x}$. Depending on this result, the update phase will either perform a classical node update as shown in Eq. (3) or a segment update.

The distance to a 1D-segment $[\mathbf{w}_i;\mathbf{w}_j]$ (see Fig. 4a) is determined via the projection point $\mathbf{p}$ of $\mathbf{x}$ onto the given segment. Let us define $\mathbf{s}_{ji}=\mathbf{w}_j-\mathbf{w}_i$ and $\mathbf{s}_{ij}=\mathbf{w}_i-\mathbf{w}_j$. Then, we may write $\mathbf{p}$ as

$$\mathbf{p}=\mathbf{w}_i+\eta_{ji}\mathbf{s}_{ji},\ 0\le\eta_{ji}\le 1 \tag{5}$$

and

$$\mathbf{p}=\mathbf{w}_j+\eta_{ij}\mathbf{s}_{ij},\ 0\le\eta_{ij}\le 1 \tag{6}$$

with $\eta_{ji}+\eta_{ij}=1$. Given the unit vectors $\hat{\mathbf{s}}_{ji}$ and $\hat{\mathbf{s}}_{ij}$, the coefficients η_{ji} and η_{ij} are given by

$$\eta_{ji}=\frac{(\mathbf{x}-\mathbf{w}_i)^T\hat{\mathbf{s}}_{ji}}{\|\mathbf{s}_{ji}\|}, \tag{7}$$

$$\eta_{ij}=\frac{(\mathbf{x}-\mathbf{w}_j)^T\hat{\mathbf{s}}_{ij}}{\|\mathbf{s}_{ij}\|}. \tag{8}$$

The squared distance $\|\mathbf{d}_{ij}\|^2$ of $\mathbf{x}$ to the 1D-segment $[\mathbf{w}_i;\mathbf{w}_j]$ is simply

$$\|\mathbf{d}_{ij}\|^2=\|\mathbf{x}-\mathbf{p}\|^2. \tag{9}$$

The 1D-segment $[\mathbf{w}_i;\mathbf{w}_j]$ that is closest to $\mathbf{x}$ is determined by

$$(\hat{i},\hat{j})=\arg\left\{\min_{ij}\|\mathbf{d}_{ij}\|\right\},\quad i,j=1,\ldots,n. \tag{10}$$

Evidently, the above equation applies only to pairs of connected nodes (i,j).

In the same way the distance to a 2D-segment (see Fig. 4b) can be determined. With $\mathbf{s}_{ji}=\mathbf{w}_j-\mathbf{w}_i$ and $\mathbf{s}_{ki}=\mathbf{w}_k-\mathbf{w}_i$ we may write $\mathbf{p}$ as

$$\mathbf{p}=\mathbf{w}_i+\eta_{ji}\mathbf{s}_{ji}+\eta_{ki}\mathbf{s}_{ki},\quad 0\le\eta_{ji},\quad \eta_{ki}\le 1,\quad \eta_{ji}+\eta_{ki}\le 1. \tag{11}$$

Similarly,

$$\mathbf{p}=\mathbf{w}_j+\eta_{ij}\mathbf{s}_{ij}+\eta_{kj}\mathbf{s}_{kj},\quad 0\le\eta_{ij},\quad \eta_{kj}\le 1,\quad \eta_{ij}+\eta_{kj}\le 1 \tag{12}$$

and

$$\mathbf{p}=\mathbf{w}_k+\eta_{ik}\mathbf{s}_{ik}+\eta_{jk}\mathbf{s}_{jk},\quad 0\le\eta_{ik},\quad \eta_{jk}\le 1,\quad \eta_{ik}+\eta_{jk}\le 1. \tag{13}$$

The coefficients η_{ji} and η_{ki} are given by

$$\eta_{ji}=\frac{(\mathbf{x}-\mathbf{w}_i)^T\hat{\mathbf{s}}_{ji}-((\mathbf{x}-\mathbf{w}_i)^T\hat{\mathbf{s}}_{ki})(\hat{\mathbf{s}}_{ki}^T\hat{\mathbf{s}}_{ji})}{\|\mathbf{s}_{ji}\|(1-(\hat{\mathbf{s}}_{ki}^T\hat{\mathbf{s}}_{ji})^2)} \tag{14}$$

$$\eta_{ki}=\frac{(\mathbf{x}-\mathbf{w}_i)^T\hat{\mathbf{s}}_{ki}-((\mathbf{x}-\mathbf{w}_i)^T\hat{\mathbf{s}}_{ji})(\hat{\mathbf{s}}_{ki}^T\hat{\mathbf{s}}_{ji})}{\|\mathbf{s}_{ki}\|(1-(\hat{\mathbf{s}}_{ki}^T\hat{\mathbf{s}}_{ji})^2)}. \tag{15}$$

Since there are only two degrees of freedom, these two coefficients determine also the other four coefficients. From simple geometrical considerations it can be shown that $\eta_{ik}=\eta_{ij}$, $\eta_{jk}=\eta_{ji}$,

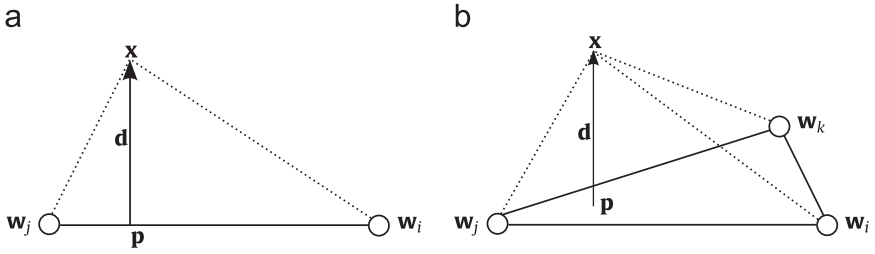

Fig. 4. (a) The projection $\mathbf{p}$ of sample point $\mathbf{x}$ onto 1D-segment $[\mathbf{w}_i;\mathbf{w}_j]$. (b) The projection $\mathbf{p}$ of sample point $\mathbf{x}$ onto 2D-segment $[\mathbf{w}_i;\mathbf{w}_j;\mathbf{w}_k]$.

$\eta_{kj}=\eta_{ki}$ and $\eta_{ij}+\eta_{ik}+\eta_{ji}+\eta_{jk}+\eta_{ki}+\eta_{kj}=2$, which allows to calculate them. The squared distance $\|\mathbf{d}_{ijk}\|^2$ of $\mathbf{x}$ to the 2D-segment (triangle) determined by $[\mathbf{w}_i;\mathbf{w}_j;\mathbf{w}_k]$ is again simply $\|\mathbf{d}_{ijk}\|^2=\|\mathbf{x}-\mathbf{p}\|^2$.

After having determined whether one of the nodes, a 1D-segment, or a 2D-segment is closest to the randomly chosen sample point, the update procedure takes place. The simplest situation is illustrated in Fig. 5a, when a single node is closest and has to be updated. This is done according to the standard SOM algorithm. In case a segment has to be updated, the nodes $[\mathbf{w}_i;\mathbf{w}_j]$ which determine the segment have to be moved such that the distance $\|\mathbf{d}_{ij}\|$ is reduced. We derive the respective movement by gradient-descent minimization on the squared segment distance. For node j we obtain

$$\begin{aligned}-\frac{1}{2}\frac{\partial \mathbf{d}_{ij}^2}{\partial \mathbf{w}_j}&=-\frac{1}{2}\frac{\partial}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})\\&=\frac{\partial \mathbf{p}^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\frac{\partial(\mathbf{w}_i+\eta_{ji}\mathbf{s}_{ji})^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\frac{\partial \eta_{ji}}{\partial \mathbf{w}_j}\mathbf{s}_{ji}^T(\mathbf{x}-\mathbf{p})+\eta_{ji}\frac{\partial \mathbf{s}_{ji}^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\eta_{ji}(\mathbf{x}-\mathbf{p})\end{aligned}\tag{16}$$

since $\mathbf{s}_{ji}$ is perpendicular to $\mathbf{x}-\mathbf{p}$ and $\partial \mathbf{s}_{ji}^T/\partial \mathbf{w}_j$ gives the identity matrix. $\partial \mathbf{p}^T/\partial \mathbf{w}_j$ denotes the matrix

$$\left(\frac{\partial \mathbf{p}^T}{\partial \mathbf{w}_j}\right)_{kl}=\frac{\partial p_l}{\partial w_k^j}.\tag{17}$$

Given the above result with the symmetry in i and j, the two displacements applied to the winner 1D-segment nodes are

$$\Delta \mathbf{w}_{\hat{i}}=\epsilon(t)\eta_{\hat{i}\hat{j}}(\mathbf{x}-\mathbf{p})\tag{18}$$

$$\Delta \mathbf{w}_{\hat{j}}=\epsilon(t)\eta_{\hat{j}\hat{i}}(\mathbf{x}-\mathbf{p}).\tag{19}$$

The movement of the two nodes is orthogonal to the line segment and illustrated in Fig. 5b.

In case a 2D-segment is closest, like in Fig. 5c, three nodes have to be updated. Gradient descent analog to above yields

$$\begin{aligned}-\frac{1}{2}\frac{\partial \mathbf{d}_{ijk}^2}{\partial \mathbf{w}_j}&=-\frac{1}{2}\frac{\partial}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})\\&=\frac{\partial \mathbf{p}^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\frac{\partial(\mathbf{w}_i+\eta_{ji}\mathbf{s}_{ji}+\eta_{ki}\mathbf{s}_{ki})^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\frac{\partial \eta_{ji}}{\partial \mathbf{w}_j}\mathbf{s}_{ji}^T(\mathbf{x}-\mathbf{p})+\eta_{ji}\frac{\partial \mathbf{s}_{ji}^T}{\partial \mathbf{w}_j}(\mathbf{x}-\mathbf{p})\\&=\eta_{ji}(\mathbf{x}-\mathbf{p}).\end{aligned}\tag{20}$$

Again, using the symmetries in i, j and k, the three displacements applied to the winner 2D-segment nodes are

$$\Delta \mathbf{w}_{\hat{i}}=\epsilon(t)\eta_{\hat{i}\hat{j}}(\mathbf{x}-\mathbf{p})=\epsilon(t)\eta_{\hat{i}\hat{k}}(\mathbf{x}-\mathbf{p})\tag{21}$$

$$\Delta \mathbf{w}_{\hat{j}}=\epsilon(t)\eta_{\hat{j}\hat{i}}(\mathbf{x}-\mathbf{p})=\epsilon(t)\eta_{\hat{j}\hat{k}}(\mathbf{x}-\mathbf{p})\tag{22}$$

$$\Delta \mathbf{w}_{\hat{k}}=\epsilon(t)\eta_{\hat{k}\hat{i}}(\mathbf{x}-\mathbf{p})=\epsilon(t)\eta_{\hat{k}\hat{j}}(\mathbf{x}-\mathbf{p}),\tag{23}$$

this time orthogonal to the triangle.

A short discussion is required concerning the segment updates. Since the displacements orthogonal to the line or triangle are not infinitesimally small but of finite size, with each update the respective line or triangle will be slightly enlarged. With many update steps, the network might increase over the borders of the data space. Several solutions to this problem are possible. The most canonical one is to add a "spring-like" term, which has to be weighted such that an appropriate shortening of the distances of the updated nodes takes place with each update step. That is, for each pair $(\mathbf{w}_i,\mathbf{w}_j)$ of nodes that are updated, the distance between them is shortened according to

$$\Delta \mathbf{w}_i=\delta_s(\mathbf{w}_j-\mathbf{w}_i)\tag{24}$$

$$\Delta \mathbf{w}_j=\delta_s(\mathbf{w}_i-\mathbf{w}_j)\tag{25}$$

with δ_s as the spring constant. The optimal magnitude of the spring constant depends on the topology as well as the number of update steps. We have chosen $\delta_s=10^{-6}$, with good results and a stable topology for 5000 iteration steps.

Another solution to the problem is to constrain the segments of the topology to constant lengths. The lengths are first calculated after the first iteration, as the topology changed from its default state to one which better approximates the anthropometric features of the tracked hand. The distances are then used to constrain neighboring nodes after an update, so that the topology does not shrink or expand. As the hand dimensions effectively remain the same in 3D space, these distances can be used regardless of hand movements.

The second solution was used for the embedded implementation which will be presented in Section 6. Good results were achieved, especially in the case of hand poses with fingers held

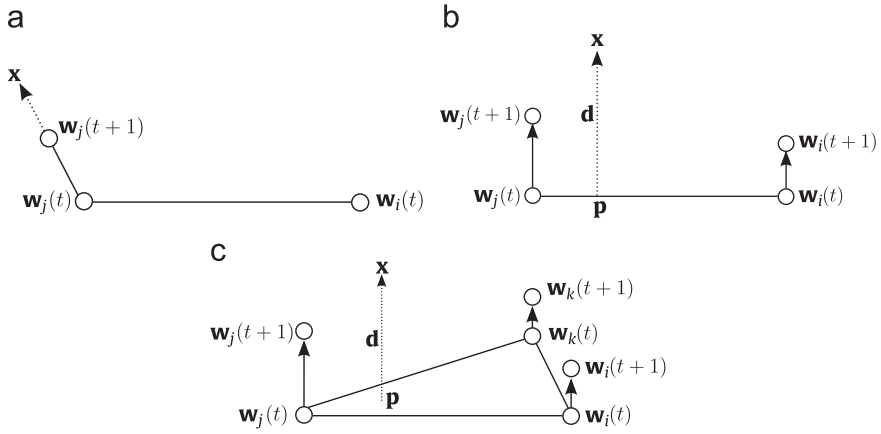

Fig. 5. (a) Weight $\mathbf{w}_j$ is displaced towards the data point $\mathbf{x}$ along the direction given by vector $\mathbf{x}-\mathbf{w}_j$. (b) Weights $\mathbf{w}_i$ and $\mathbf{w}_j$ are displaced towards the data point $\mathbf{x}$ along directions parallel to the vector formed by $\mathbf{x}$ and its projection $\mathbf{p}$. (c) Weights $\mathbf{w}_i$, $\mathbf{w}_j$ and $\mathbf{w}_k$ are displaced towards the data point $\mathbf{x}$ along directions parallel to the vector formed by $\mathbf{x}$ and its projection $\mathbf{p}$. The closer $\mathbf{p}$ is to a node, the larger its update.

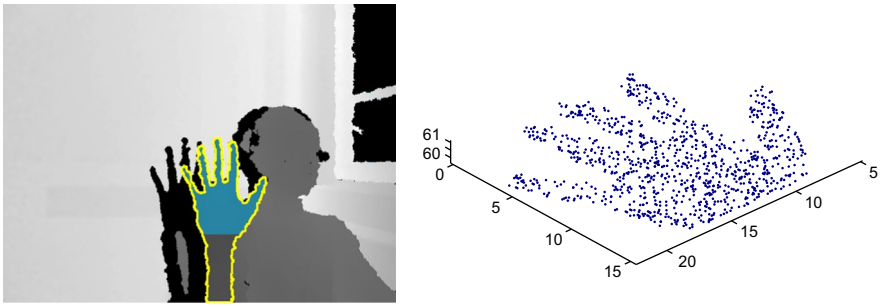

Fig. 6. Starting from a frame captured with the Kinect, we extract the hand based on the closest object assumption. We remove the points corresponding to the forearm and obtain the segmented hand and the 3D point cloud used for the tracker's learning stage.

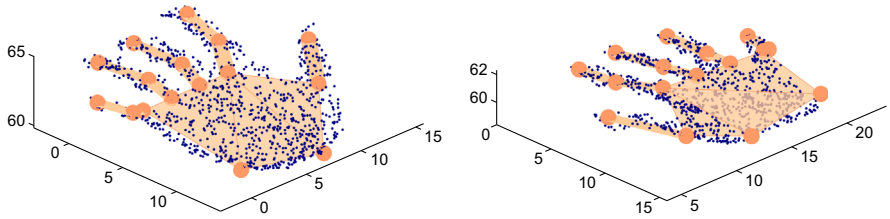

Fig. 7. The extended SOM tracker converges to the open palm topology, for both left and right hands.

close together: as the topology segments remain constant, the fingers are kept straight and do not bend or shrink, as they did with the standard SOM algorithm implementation.

5.2. Results and evaluation

Our tracking algorithm uses a Kinect device (Fig. 9b) for 3D data acquisition. The points corresponding to the hand are extracted with a threshold segmentation performed on the given depth frame, based on the assumption that the hand is the closest object to the camera. The removal of forearm points is performed based on morphological erosion, followed by detection of the center of the palm, using the distance transform. This yields the point cloud our algorithm works on (Fig. 6).

In the beginning of the tracking procedure we first have to initialize the network. This is done by translating the open-hand topology to the center of the hand point cloud. The user also has to present an open hand that is pointing upward. In its default state, the topology size is that of an average hand, but it adapts to the user's hand quickly, after the first few frames. This way, the match does not have to be precise, and the SOM will conform to the hand even if the fingers are not held completely separated or if the hand is not completely straight. The extended SOM algorithm is applied to each individual frame, each time with 5000 training steps (a training step consists of a random choice of a data point, followed by a competition and update step), with the following frames using the previous position of the SOM topology as a starting point.

In each new frame, we perform a training during which the network receives as input the data points from the given hand point cloud. Fig. 7 shows how the hand network has converged into the point cloud of a given frame, for the left and right hand respectively. As desired, the network represents the given hand topology by minimizing the mean squared distance between the data points and the network with its 1D and 2D segments.

Fig. 8 shows the results obtained with the extended SOM algorithm on a recorded dataset of a number of gestures — open palm, four extended fingers (index, middle, ring, little) and three extended fingers (thumb, index, middle). For a qualitative evaluation of the algorithm we have extracted four frames — the series of pictures in the left column — showing the convergence of the topology. We have then assessed the quantitative performance of the algorithm as the error between the fingertip nodes of the topology and manually labeled fingertips on the dataset. We show this in the right column, as the root mean square error in centimeters over time.

6. Embedded implementation

As mobile platforms shrink in size, gestural interfaces will start to become a viable alternative to touch-based interaction — the user can only do so much with a limited amount of touchscreen real estate, before his fingers start to get in the way. Although hands-free gesture control is not a new technology, it has seen little use in mobile computing. Almost every commercial software framework on the market today is aimed at desktop PCs, due to the computing power required by the algorithms they use.

Commercially, there are a number of solutions for desktop PCs, such as the Omek Grasp and Beckon,[4] the SoftKinetic iisu SDK[5] and the Microsoft Kinect for Windows,[6] which work with existing 3D cameras and provide a framework for body tracking and gesture recognition.

Probably the most widely known body tracking solution is the Microsoft Kinect for Xbox360. It employs machine learning algorithms to estimate the user's body posture [21], which is then used as input for interactive games on the console.

Regarding hand pose estimation, a recent collaboration between Intel, Creative and SoftKinetic released the Creative Interactive Gesture Camera Developer Kit.[7] It is a near-range time-of-flight camera that allows tracking of the user's hand up to 1 m. While it does not fully model the hand in 3 dimensions, it does provide the extended fingertips' position and a few anatomical landmarks (palm, elbow).

[4] http://www.omekinteractive.com/solutions/.
[5] http://www.softkinetic.com/en-us/solutions/iisusdk.aspx.
[6] http://www.microsoft.com/en-us/kinectforwindows/.
[7] http://www.click.intel.com/intelsdk/.

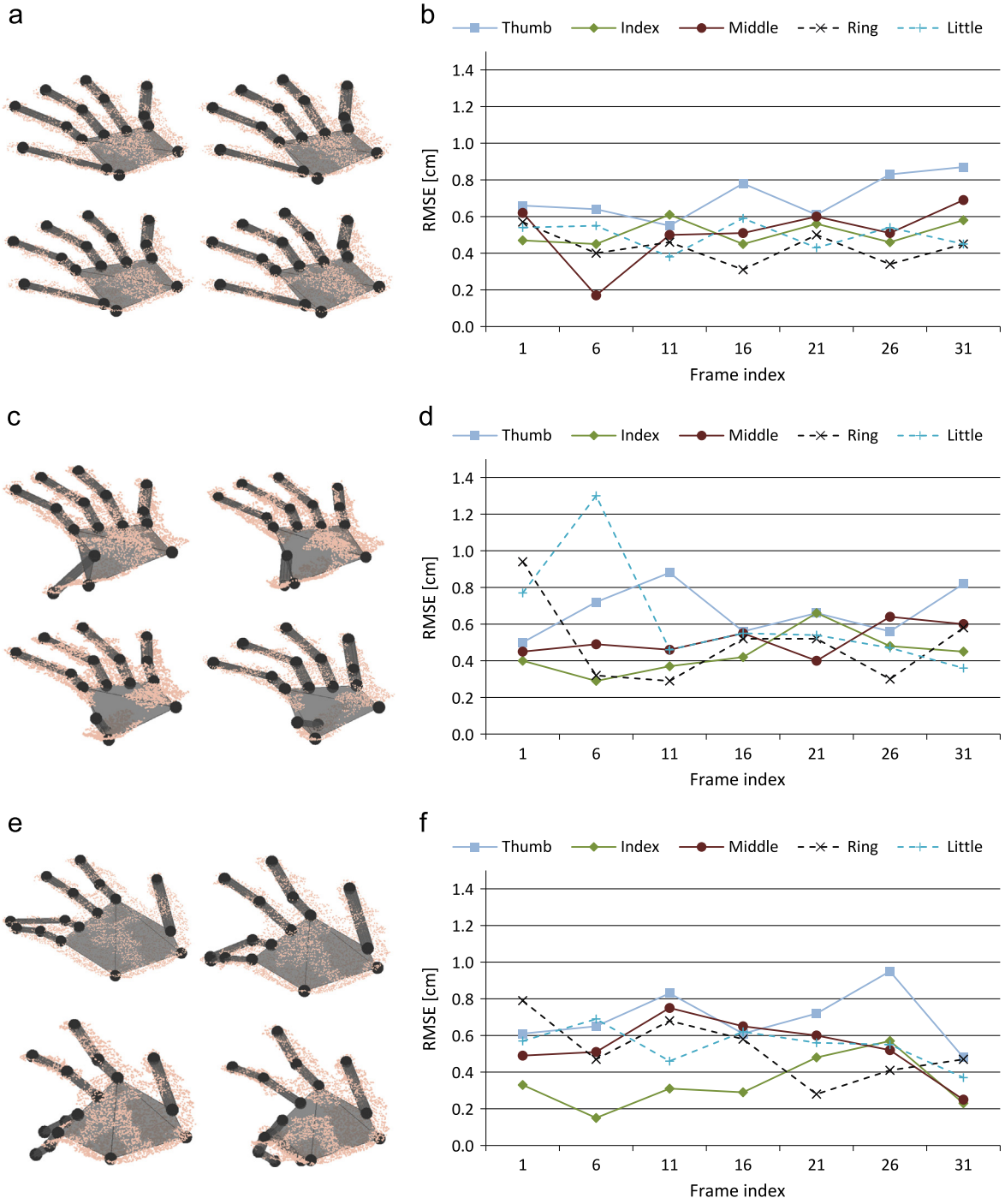

Fig. 8. Qualitative (left column) and quantitative (right column) assessment of the extended SOM topology on various hand poses: open hand (a,b); four extended fingers (c,d); and three extended fingers (e,f).

6.1. Hardware

The hardware that is used in our implementation of the above algorithms consists of two main parts: the ARM development platform and the 3D camera, which provides input data for the algorithm. The resulting images are displayed on a standard computer monitor.

As an embedded platform suitable for mobile computing we have chosen the PandaBoard ES (Fig. 9a), which is the next iteration of the popular Pandaboard platform. It is powered by a Texas Instruments OMAP4460 system-on-chip (SoC). The processor itself is used in a number of mobile devices available on the market such as the Samsung Galaxy Nexus. The board features a 1.2 GHz dual-core Cortex-A9 ARM CPU, a PowerVR SGX 540 graphics processing unit, 1 GB DDR2 SDRAM, two USB 2.0 ports, Ethernet connection and various other peripherals.

The operating system of choice is Linux. We are using a Linaro distribution, which is built and optimized for the ARM architecture implemented in OMAP4460, with the standard toolchain installed.

3D cameras have historically been either too expensive or of too low quality to provide an accurate representation of the surrounding environment. This changed when Microsoft unveiled in 2010 the Kinect, a peripheral device for the Xbox360 gaming

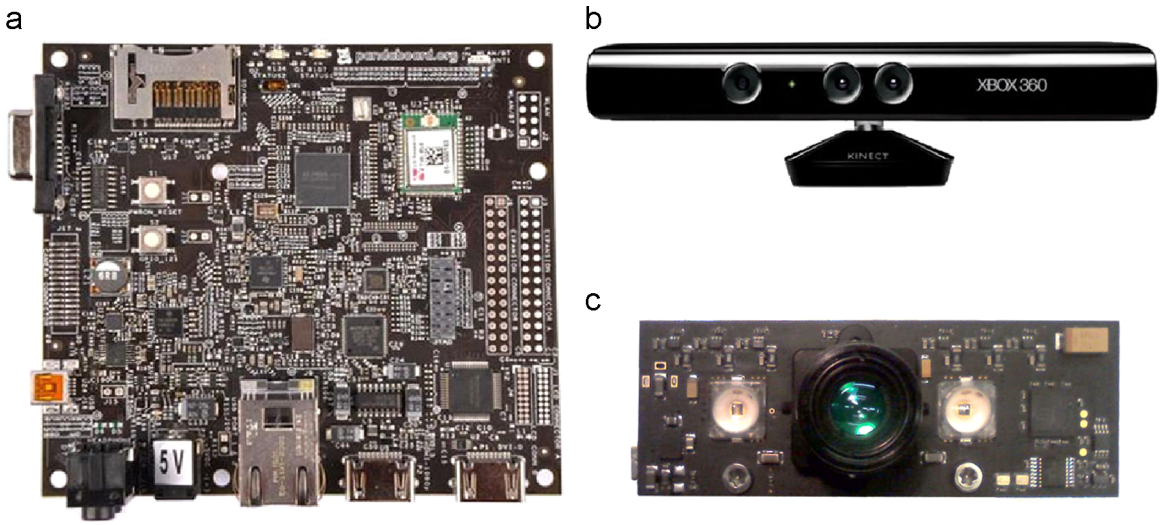

Fig. 9. (a): The embedded platform used; (b) and (c) the two cameras used.

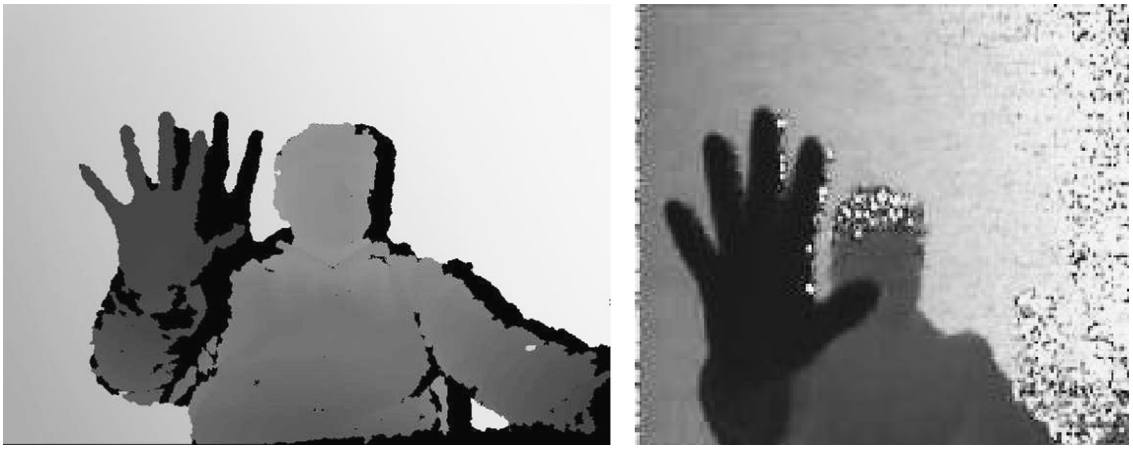

Fig. 10. Depth stream comparison: Microsoft Kinect (left) and PMD CamBoard Micro (right).

console. The Kinect is able to provide reliable 3D data of up to 15 m at the same time with (horizontally offset) RGB data. Although the native resolution of the device is 640 by 480 pixels, its spatial resolution is about 4 times lower, varying with depth and object geometry [22]. A comparable time-of-flight (TOF) 3D camera at that time was available at a price 20 times higher than Microsoft's device. We have chosen the Kinect as the main camera in this implementation, together with a multi-platform community developed driver, as there are no official drivers for the Linux operating system used on the PandaBoard. Depth is provided with 11 bits of precision, although beyond 4 m the accuracy lowers significantly. The minimum distance is 40 cm, at which point the reflected light starts saturating the infrared camera. Although detail varies with distance and object geometry [22], we have found that there is enough resolution in the depth map provided by the Kinect between 40 and 100 cm so that the fingers are separated, and the hand can be recognized.

The second device used is a time-of-flight camera, the PMD CamBoard. It can output a depth stream at a resolution of 207×204 pixels and 30 fps. Due to the small form factor of the camera ($85 \times 35 \times 40$ mm) and subsequent size of the LEDs, the distance is limited, accurate readings being made from 20 cm up to 150 cm. The driver for the camera can provide a *flag* image for each frame, in which invalid or low signal pixels are marked and can be used to filter the depth map so that only valid data will be used.

A comparison between the two cameras is shown in (Fig. 10). On the left, the Kinect provides a large but blotchy-looking image. The black areas are shadows, objects that do not reflect infrared well, or areas where the depth computation algorithm failed to produce results. On the right, the PMD CamBoard provides a small and blurred image. At large depths the signal is too weak to be captured back by the sensor, which produces large amounts of noise. There is minimal shadowing due to the LED lights, which are placed closer to the sensor compared to the Kinect. An image artifact of the PMD camera is the thickening of close objects due to the light saturating the sensor. In order to overcome this, the integration time can be adjusted so that the sensor captures less light, but as a direct consequence the maximum distance is reduced.

6.2. Implementation description

In contrast to other algorithms, which use only depth maps (usually as a pre-processing step for easier background segmentation) [2], the SOM approach needs the actual 3D point cloud, which contains the coordinates in space of every pixel in the image, not just the distance from the camera. While this is done

automatically by the driver of the PMD CamBoard, the Kinect only provides the depth map. As the intrinsic camera properties (field of view, focal length, pixel size) are known, one can calculate the 3D positions of the points in the 2D depth map.

Because the method requires only the points in 3D space that belong to the subject, they must be separated from the environment background. This is achieved by clipping the points outside a bounding box for the body, and choosing the closest object to the camera that is bigger than a threshold size in the case of the hand. We have used the expanded SOM topology for the body and hand and also the extended SOM model that has been described in the previous section.

In order to be able to run this algorithm in real-time (30 fps or more) on the embedded system, a number of optimizations have been made. We aimed to access memory as little as possible, to prevent the large overhead penalty, and keep most data in the cache. The PandaBoard also features a single-instruction-multiple-data (SIMD) pipeline that can significantly speedup computations. Repetitive operations on large data can be significantly accelerated using the NEON engine. By filling up the pipeline, data can be effectively processed in parallel, giving a performance boost to the algorithm. For non-critical data, in order not to incur the performance penalty of floating point computations, we are using fixed-point arithmetic.

One notable feature of the algorithm with regards to performance is that its complexity does not depend on the input data. Indeed, regardless of the size of the image or the point cloud, the algorithm will only process as many points as there are iterations. For example, in the case of the standard SOM algorithm, if a topology is updated for 5000 iterations, it will not matter if the hand point cloud has 500 or 50,000 points, there will always be only 5000 points that will be randomly chosen from the cloud for the competition/update phase. This means that for the smaller point cloud there will be points chosen multiple times, and for the larger one there will be points that are not chosen at all, but since the algorithm is stochastic in nature, the points will be chosen uniformly and the topology will be able to fit to the point cloud.

The performance of the standard SOM algorithm is thus related only to the parameters of the learning process: the number of iterations, the number of nodes in the topology and the maximum number of neighbors that a node can have (e.g. 4, in the case of the 44-node topology shown in Fig. 1c). This is a useful feature to have as the performance (i) will not depend on the input data and (ii) can be throttled to a required computational load for lower-power systems. Computationally, the algorithm makes use only of simple operations, like addition and subtraction. The use of a square root operation is also not required as the distance is only used in comparisons.

In the case of the extended SOM, the computational load is increased, as distances are calculated in a more complex way for segments and planes and require the use of more complex operations like division and square root. Nevertheless, the overall complexity does not depend on the input data as well and can be adjusted by changing the learning parameters to suit the application.

Of course, the complexity of supporting algorithms has to be considered, too, which might depend on the input data size, such as the segmentation algorithm used to get the hand point cloud prior to the SOM.

6.3. Results

After implementing the standard SOM for the whole-body and the human-hand skeleton, we have obtained the results shown in Figs. 2 and 3. After all the optimizations described in the previous section have been implemented, we have been able to successfully reach our target of real-time performance, at 30 frames per second.

The results for the extended SOM can be seen in Figs. 11 and 12. To solve the problem of expanding segments we have used the

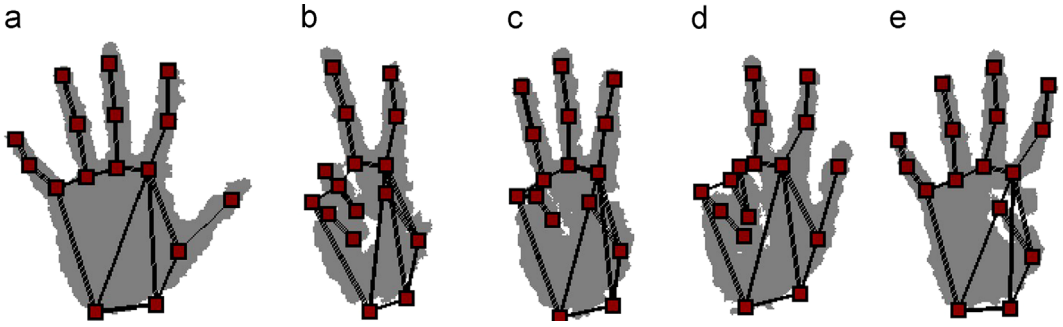

Fig. 11. The extended SOM results for various hand poses.

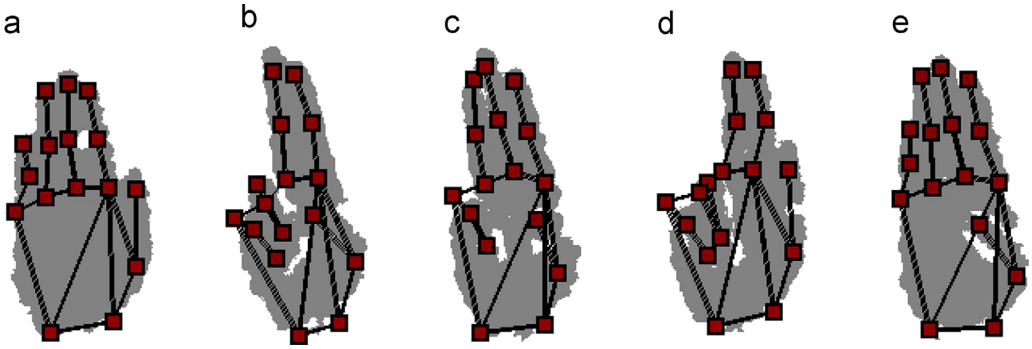

Fig. 12. The extended SOM results for difficult hand poses.

segment length constraint approach, as described in Section 5.1, with good results. Although the competition and update phases are more complex than those of the standard SOM, the algorithm still runs in real time with the same number of iterations, as the new topology has less than half the nodes of the old one (16 vs. 37). In Fig. 11 we show five hand poses taken directly from the real-time video on the embedded platform. The qualitative performance is similar with the one of the node-only SOM implementation. The topology converges correctly on the straightened as well as the bent fingers.

In Fig. 12, the corresponding poses in which the straight fingers are held together are shown. The fingers remain in the correct places and do not scatter — the new segment-plane updates solve the problem of the previous SOM implementation that appeared when data points were too close to each other and the nodes from one finger wandered into the space of other fingers. This allows for a more robust representation of the hand gestures, as melded fingers (that could come from out-of-plane hand rotations or hand which are too far away for the cameras to distinguish between fingers) are no longer a problem.

7. Conclusion

In this paper, we have presented a self-organizing map algorithm that can be used to track the full body and hand skeleton, in real time. The algorithm is able to produce a robust estimate of the human body and hand pose, at comparatively low computational cost.

We also proposed an extension to the standard SOM algorithm in which we represent the data cloud not only with the nodes of the network but also with the line segments and planes between the nodes. This approach allows an improved representation and tracking of the hand pose due to its anatomical fidelity compared to the standard SOM topology. The tracker can be further improved by adding anatomical and anthropometric constraints to the hand model, such as limiting the size of the palm or the degree of movement of the fingers and finger joints.

The computational efficiency of the method makes it ideal for implementation on a low-powered system such as an embedded platform. We have implemented both the standard and the extended SOM algorithm, with good qualitative results, as well as real-time performance on a Pandaboard ES system. Our algorithm could be used in embedded devices, which need low-power, low-complexity solutions to enable gesture technologies – granting extended interaction capabilities to current and future mobile interfaces. Natural user interfaces can be used to enhance the usability of devices ranging from the current mobile devices to the next generation head-mounted displays.

From the testing done with the Microsoft Kinect and PMD CamBoard we concluded that the method is robust and can adapt to any 3D data that is being supplied, as long as it is accurate enough, meaning that the proposed algorithm is able to work with a wide range of cameras, possibly the ones that will be used in the next generation of gesture-enabled mobile computing interfaces. Another benefit is that the self-organizing map approach can easily be extended to any deformable object that needs to be tracked by simply changing the network topology. This presents a definite advantage over methods that use machine learning to recognize the objects, as the self-organizing map algorithm need not be trained in advance.

Acknowledgments

This research is supported by the German Ministry of Education and Research (BMBF) under grant number 01IS10049B, the EXIST program of the German Ministry of Economics and Technology (BMWi), and by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/1]. A. State would like to thank for the support by The German Academic Exchange Service Programme "Ostpartnerschaften", the University of Lübeck and The Sectoral Operational Programme Human Resources Development 2007–2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement POSDRU/86/1.2/S/61756.

References

[1] G. ElKoura, K. Singh, Handrix: animating the human hand, in: SCA, 2003, pp. 110–119.
[2] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, X. Twombly, Vision-based hand pose estimation: a review, Comput. Vis. Image Underst. 108 (1–2) (2007) 52–73, http://dx.doi.org/10.1016/j.cviu.2006.10.012, ISSN 1077-3142. URL ⟨http://www.sciencedirect.com/science/article/pii/S1077314206002281⟩, Special Issue on Vision for Human–Computer Interaction.
[3] A. State, F. Coleca, E. Barth, T. Martinetz, Hand tracking with an extended self-organizing map, in: P.A. Estvez, J.C. Prncipe, P. Zegers (Eds.), Advances in Self-Organizing Maps, Advances in Intelligent Systems and Computing, vol. 198, Springer, Berlin, Heidelberg, 2013, pp. 115–124, http://dx.doi.org/10.1007/978-3-642-35230-0_12, ISBN 978-3-642-35229-4 ⟨http://dx.doi.org/10.1007/978-3-642-35230-0_12⟩.
[4] M. Haker, M. Böhme, T. Martinetz, E. Barth, Self-organizing maps for pose estimation with a time-of-flight camera, in: Proceedings of the DAGM Workshop on Dynamic 3D Imaging, Lecture Notes in Computer Science, vol. 5742, 2009, pp. 142–153. ⟨http://www.springerlink.com/content/006305183070t383/⟩.
[5] T. Kohonen, Self-organizing maps, in: Springer Series in Information Sciences, 1995.
[6] E. Foxlin, Motion tracking requirements and technologies, in: Handbook of Virtual Environments: Design, Implementation, and Applications, Lawrence Erlbaum Associates, 2002, pp. 163–210.
[7] T.B. Moeslund, A. Hilton, V. Krüger, A survey of advances in vision-based human motion capture and analysis, Comput. Vis. Image Underst. 104 (2–3) (2006) 90–126.
[8] J. Regh, T. Kanade, Visual tracking of high DOF articulated structures: an application to human hand tracking, in: J.-O. Eklundh (Ed.), Proceedings of 3rd European Conference on Computer Vision, Springer-Verlag, 1994, pp. 35–46.
[9] J. Segen, S. Kumar, Shadow gestures: 3D hand pose estimation using a single camera, in: Proceedings of Conference on Computer Vision and Pattern Recognition, 1999.
[10] C.-C. Chang, Adaptive multiple sets of {CSS} features for hand posture recognition, Neurocomputing 69 (16–18) (2006) 2017–2025.
[11] C.-C. Chang, C.-Y. Liu, W.-K. Tai, Feature alignment approach for hand posture recognition based on curvature scale space, Neurocomputing 71 (10–12) (2008) 1947–1953.
[12] R. Rosales, V. Athitsos, S. Sclaroff, 3D hand pose reconstruction using specialized mappings, in: Proceedings of International Conference on Computer Vision, vol. 1, 2001, pp. 378–385.
[13] V. Athitsos, S. Sclaroff, Estimating 3D hand pose from a cluttered image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2003.
[14] B. Stenger, A. Thayananthan, P.H.S. Torr, R. Cipolla, Hand pose estimation using hierarchical detection, in: Proceedings of International Workshop Human–Computer Interaction, 2004, pp. 105–116.
[15] M. Bray, E. Koller-Meier, L.V. Gool, Smart particle filtering for 3D hand tracking, in: Sixth IEEE International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society, Los Alamitos, CA, USA, 2004, p. 675.
[16] M. Haker, M. Böhme, T. Martinetz, E. Barth, Deictic gestures with a time-of-flight camera, in: Gesture in Embodied Communication and Human–Computer Interaction – International Gesture Workshop, 2009.
[17] N. Shimada, K. Kimura, Y. Shirai, Real-time 3D hand posture estimation based on 2D appearance retrieval using monocular camera, in: Proceedings of IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. ISSN 1530-1044, pp. 23–30. http://dx.doi.org/10.1109/RATFG.2001.938906, 2001.
[18] M. Bayazit, A. Couture-Beil, G. Mori, Real-time motion-based gesture recognition using the GPU, in: Proceedings of the IAPR Conference on Machine Vision Applications, 2009, pp. 9–12.
[19] T. Bierz, A. Ebert, J. Meyer, GPU accelerated gesture detection for real time interaction, in: Visualization of Large and Unstructured Data Sets'07, 2007, pp. 64–75.
[20] I. Oikonomidis, N. Kyriazis, A.A. Argyros, Efficient model-based 3D tracking of hand articulations using kinect, in: British Machine Vision Conference, vol. 2, Dundee, UK, 2011.
[21] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, Real-time human pose recognition in parts from single depth images, in: Computer Vision and Pattern Recognition, 2011, URL ⟨http://research.microsoft.com/apps/pubs/default.aspx?id=145347⟩.

[22] M. Andersen, T. Jensen, P. Lisouski, A. Mortensen, M. Hansen, T. Gregersen, P. Ahrendt, Kinect Depth Sensor Evaluation for Computer Vision Applications – Technical Report ECE-TR-6, Department of Engineering Electrical and Computer Engineering, Aarhus University, Denmark, 2012.

Foti Coleca earned his master's degree in 2011 from Politehnica University of Bucharest, specializing in digital image processing. He is currently a Ph.D. student at the Institute for Neuro- and Bioinformatics and collaborating with the Neurosurgery department of the University of Lubeck and the tech startup gestigon GmbH. His current research focuses on gesture recognition with 3D sensors and medical applications of gestural interfaces.

Andreea State gained a master in digital imaging and bioinformatics from University "Politehnica" of Bucharest in 2012. She completed her dissertation thesis on the topic of hand tracking using Self-Organizing Maps as a collaboration between the University of Lübeck and University "Politehnica" of Bucharest. Her main interests are image processing, machine learning and neural networks.

Sascha Klement holds a master in computer science with focus on neuroinformatics, machine learning and image processing. Since 2006, as a research associate at the Institute for Neuro- and Bioinformatics at the University of Lübeck and as a project leader at the Pattern Recognition Company GmbH, he has managed and carried out numerous research and development projects in the area of industrial automation and image processing, pattern recognition, time-of-flight technology and human machine interfaces. Since 2011, as a managing director and CTO of gestigon, Sascha Klement is responsible for the technical roadmap, algorithm design and software architecture.

Erhardt Barth received the Ph.D. degree in electrical and communications engineering from the Technical University of Munich, Munich, Germany. He is a Professor at the Institute for Neuro- and Bioinformatics, University of Lübeck, Lübeck, Germany, where he leads the research on human and machine vision. He has conducted research at the Universities of Melbourne and Munich, the Institute for Advanced Study in Berlin, and the NASA Vision Science and Technology Group in California. Dr. Barth is an associate editor of the IEEE Transactions on Image Processing.

Thomas Martinetz is full professor of computer science and director of the Institute for Neuro- and Bioinformatics. He studied physics at the TU München and obtained his doctoral degree in Biophysics at the Beckman Institute for Advanced Science and Technology of the University of Illinois at Urbana-Champaign. From 1991 to 1996 he led the project Neural Networks for automation control at the Corporate Research Laboratories of the Siemens AG in Munich. From 1996 to 1999 he was Professor for Neural Computation at the Ruhr-University of Bochum and head of the Center for Neuroinformatics.