Image and Vision Computing 32 (2014) 904–915

Contents lists available at ScienceDirect

Image and Vision Computing

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

Partial least squares-based human upper body orientation estimation with combined detection and tracking ☆

I. Ardiyanto *, J. Miura

Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Aichi 441-8580, Japan

ARTICLE INFO

Article history:
Received 30 October 2013
Received in revised form 8 March 2014
Accepted 4 August 2014
Available online 11 August 2014

Keywords:
Human upper body orientation
Partial least squares
Multi-level HOG-LBP
Random Forest classifier
UKF tracker

ABSTRACT

This paper deals with the problem of estimating the human upper body orientation. We propose a framework which integrates estimation of the human upper body orientation and the human movements. Our human orientation estimator utilizes a novel approach which hierarchically employs partial least squares-based models of the gradient and texture features, coupled with the random forest classifier. The movement predictions are done by projecting detected persons into 3D coordinates and running an Unscented Kalman Filter-based tracker. The body orientation results are then fused with the movement predictions to build a more robust estimation of the human upper body orientation. We carry out comprehensive experiments and provide comparison results to show the advantages of our system over the other existing methods.

1. Introduction

Human body orientation estimation is one of challenging yet useful tasks for the mobile robot and surveillance applications. The human body orientation can tell us how people interact with each other in the surveillance scenes. For example, we may predict that a group of persons facing each other for a long time are having conversation, or other social semantic predictions such as waiting for the bus together.

From the mobile robots point of view, the human body orientation can assist the robot to get a better prediction for avoiding a person when doing navigation tasks. It also helps the robot to make a social interaction with the human in outdoor navigation, such as approaching a person and asking the way. Here the robot certainly needs the estimation of human orientation for facing the person.

Several works try to accomplish the human body orientation problem. Andriluka et al. [3] use banks of viewpoint specific part based detectors and linear Support Vector Machine (SVM) for estimating the whole body orientation. Another recent work is done by Weinrich et al. [4] which performs the human upper body orientation estimation using Histogram of Oriented Gradient (HOG) features and SVM Tree. A work by Baltieri et al. [21] employ a Mixture of Approximated Wrapped Gaussian (MoAWG) weighted by the detector outputs for increasing the correct estimation rates. The above researchers do not consider how the person movements will affect the overall estimation results.

One notable work by Chen et al. [5] uses multi-level HOG features and sparse representation for classifying the human pose. They also employ a soft-coupling technique between the whole body orientation and its velocity using the particle filter framework.

To solve the problems we have mentioned above, here we propose a system for detecting and estimating the human upper body orientation, as well as its motion. We prefer to use the upper body part rather than the whole body for gaining the robustness under occlusion; the full body is often partially occluded by small objects such as chair, table, bicycle, and so on.

Our main contribution resides in the use of the partial least squares (PLS)-based model of gradient and texture features for estimating the human upper body orientation. Here, our PLS model is a modification of the one used in [16] which has been successfully applied for human detection. We also provide an Unscented Kalman Filter (UKF) framework integrating the human movement prediction and the orientation estimation for improving the estimation results.

We organize the paper as follows. First we describe the detection and estimation of human upper body orientation using partial least squares models in Section 2. Section 3 explains the integration of the orientation estimation results and the tracking. We then provide the comparison of several methods and the result of extensive experiments in Section 4. Lastly, we conclude our work and give some possible future works.

☆ This paper has been recommended for acceptance by Vassilis Athitsos.

* Corresponding author.
E-mail addresses: iardiyanto@aisl.cs.tut.ac.jp (I. Ardiyanto), jun.miura@tut.jp (J. Miura).

http://dx.doi.org/10.1016/j.imavis.2014.08.002
0262-8856/

Fig. 1. Eight classes of the human upper body orientation, representing (from left to right) front, front-left, left, back-left, back, back-right, right, and front-right directions.

2. Model-based human upper body orientation estimation

2.1. Hierarchical system

Our system is built in a hierarchical manner. First we detect and create bounding boxes around the human upper body using a fast detector. These detection results are then fed to the orientation estimator part. We divide the orientation into eight quantization (see Fig. 1). Results from the detector and the orientation estimator are used as the observation inputs for the tracker. Fig. 2 explains our framework on the human upper body orientation estimation as described above.

Our system is composed of the image-based orientation estimation and the tracking system (see Fig. 2). The influence between the systems is one-directional. That is, the orientation estimation system runs independently, and its results are used by the tracking system for increasing the robustness of its pose and motion estimation. This also implies that the orientation estimation system can be applied to still images.

2.2. Human upper body detection

Histogram of Oriented Gradients (HOG) [1] is one of state-of-the-art descriptor for the whole body person detection. However, the original HOG algorithm is slow and unsuitable for the real time applications. Here we exploit the extended work of HOG by [12], which employs Adaboost for selecting features and cascade rejection for speeding up the detection time. Instead of using the whole human body, we prefer to exploit the upper body part for gaining the robustness under occlusion, as we will show in Section 4.2. The detection results (bounding boxes of the human upper body) are then used as the input for the orientation estimator part.

2.3. Extracting features for human upper body orientation estimation

The other works (e.g. [4] and [5]) solely depend on the gradient features, which capture the body shape. For the human upper body orientation case, we make an assumption that using only the body shape is not enough, for example, there is no big difference between the shape of the body facing front and back. Therefore, we propose a combination of the gradient-based and texture-based features which grab the shape and the texture cues. Here we expect the textured part of the body such as the face will be captured. We then apply the modified partial least squares (PLS) models for enhancing the important cues of the human orientation.

2.3.1. Shape cue

For capturing the human upper body shape, we use a multi-level HOG descriptor [1]. The gradient magnitude for each upper body image sample is first computed using 1-D mask $[-1\ 0\ 1]$ on each x and y direction. Every sample is divided into 3×4, 6×8, and 12×16 blocks, where each block consists of four cells. The gradient orientation is then quantized into nine bins. Overall we have 252 blocks and 9072 dimensional feature vectors of HOG descriptor.

2.3.2. Texture cue

We use Local Binary Pattern (LBP), adopting the work of [2], to make a texture descriptor. We calculate image textures using $\mathrm{LBP}_{8,1}$ operator for each pixel

$$I_{\mathrm{LBP}c} = \sum_{p=0}^{7} 2^p \varsigma\left(I_p - I_c\right), \quad \varsigma(a) = \begin{cases} 1 & \text{for } a \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where I_c is the center pixel from which we calculate the LBP value $I_{\mathrm{LBP}c}$ and p is eight-surrounding pixels of I_c. We then divide the LBP

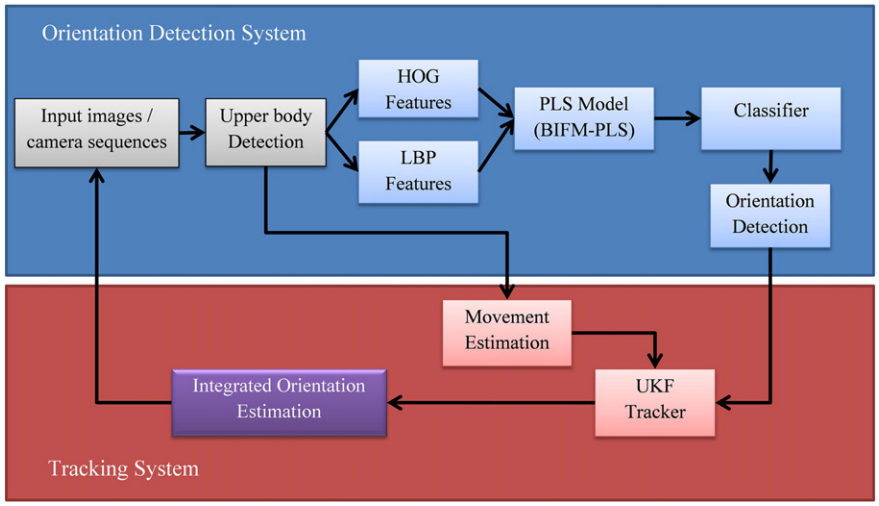

Fig. 2. Diagram of the human upper body orientation estimation system.

images into multiple blocks similar to the HOG features above. For each block, we make a histogram containing 59 labels based on uniform patterns.[1] This procedure gives us 14,868 dimensional feature vectors in total.

2.4. Partial least squares for modeling features

2.4.1. Partial least squares

Partial least squares (PLS) is a statistical method used for obtaining relations between sets of observed variables through the estimation of a low dimensional latent space which maximizes the separation between samples with different characteristics []. The PLS builds new predictor variables called latent variables, as combinations of a matrix $\boldsymbol{X}$ of the descriptor variables (features) and a vector $\boldsymbol{y}$ of the response variables (class labels).

Let us consider a problem with γ samples, $\boldsymbol{X} \subset \mathbb{R}^{\delta}$ be an δ-dimensional space representing the feature vectors and $y \subset \mathbb{R}$ denote a 1-dimensional space of the class labels. The PLS then decomposes the $(\gamma \times \delta)$ matrix of zero mean variables $\boldsymbol{X}$ and the vector of zero mean variables y into

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E}, \\ \boldsymbol{y} &= \boldsymbol{U}\boldsymbol{q}^T + \boldsymbol{r}, \end{aligned} \tag{2}$$

where $\boldsymbol{T}$ and $\boldsymbol{U}$ represent $(\gamma \times s)$ matrices of s extracted latent vectors, the $(\delta \times s)$ matrix $\boldsymbol{P}$ and the $(1 \times s)$ vector $\boldsymbol{q}$ are the loadings, and the $(\gamma \times \delta)$ matrix $\boldsymbol{E}$ and the $(\gamma \times 1)$ vector $\boldsymbol{r}$ are the residuals.

Here we implement the nonlinear iterative partial least squares (NIPALS) algorithm [17] to find a set of projection vectors (weight vectors) $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_s\}$such that

$$[cov(\boldsymbol{t}_i, \boldsymbol{u}_i)]^2 = \max_{|\boldsymbol{w}_i|=1} [cov(\boldsymbol{X}\boldsymbol{w}_i, \boldsymbol{y})]^2, \tag{3}$$

where $\boldsymbol{t}_i$ is the i-th column of matrix $\boldsymbol{T}$, $\boldsymbol{u}_i$ is the i-th column of matrix $\boldsymbol{U}$, and $cov(\boldsymbol{t}_i, \boldsymbol{u}_i)$ is the sample covariance between latent vectors $\boldsymbol{t}_i$ and $\boldsymbol{u}_i$. The process of constructing projection vectors $\boldsymbol{W}$ is shown in Algorithm 1.

Algorithm 1. PLS/NIPALS algorithm

procedure PLS()
  Init $\boldsymbol{X}_0, y_0$
  for $i = 1 \rightarrow s$ **do**
    $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{X}_i^T y_i$
    $\boldsymbol{w}_{i+1} \leftarrow \frac{\boldsymbol{w}_{i+1}}{\|\boldsymbol{w}_{i+1}\|}$
    $\boldsymbol{t}_{i+1} \leftarrow \boldsymbol{X}_i \boldsymbol{w}_{i+1}$
    $\boldsymbol{P}_{i+1} \leftarrow \frac{\boldsymbol{X}_i^T \boldsymbol{t}_{i+1}}{\boldsymbol{t}_{i+1}^T \boldsymbol{t}_{i+1}}$
    $q_{i+1} \leftarrow \frac{y_i^T \boldsymbol{t}_{i+1}}{\boldsymbol{t}_{i+1}^T \boldsymbol{t}_{i+1}}$
    $\boldsymbol{X}_{i+1} \leftarrow \boldsymbol{X}_i - \boldsymbol{t}_{i+1}\boldsymbol{P}_{i+1}^T$
    $y_{i+1} \leftarrow y_i - \boldsymbol{t}_{i+1} q_{i+1}$
  end for
end procedure

2.4.2. Block Importance Feature Model

Here we introduce our PLS-based feature models, Block Importance Feature Model of PLS (BIFM-PLS). We also provide another simple PLS model called Combined Feature Model of PLS (CFM-PLS) for comparison purpose.

The Combined Feature Model of PLS (CFM-PLS) is created by concatenating all of HOG–LBP features into one vector, and simply run the PLS algorithm on it with specified number of the latent vectors. As the result, the CFM-PLS produces a reduced set of features by projecting the feature vectors $\boldsymbol{f} \subset \boldsymbol{X}$ onto the weight vectors,

$$\breve{\boldsymbol{x}} = \boldsymbol{W}\boldsymbol{f}. \tag{4}$$

This result is then used as the input for the classifier. Fig. 3 explains the procedure of building CFM-PLS.

The CFM-PLS method, which resembles an ordinary PLS technique, is only used for comparison purposes, and from now we focus on the BIFM-PLS method.

The Block Importance Feature Model of PLS (BIFM-PLS) is built by an idea that not all of blocks or features have a high contribution to the classification. Here we want to examine the contribution of each block and discard the one which has low importance. Unlike the CFM-PLS which highly reduces the feature space, the BIFM-PLS is intended to retain some details about the features to be fed up to the classifier.

We adopt and extend the method of [16] for picking out the representative blocks. For creating BIFM-PLS, we employ the feature selection called Variable Importance on Projection (VIP) (see [16] and [18]). The VIP gives a score for each feature representing its predictive power in the PLS model. The VIP of the i-th feature $\boldsymbol{f}$ is given by

$$VIP_i(\boldsymbol{f}) = \sqrt{\frac{\kappa \sum_{j-1}^{p} b_j^2 \boldsymbol{w}_{ij}^2}{\sum_{j=1}^{p} b_j^2}}, \tag{5}$$

where κ is the number of features, $\boldsymbol{w}_{ij}$ denotes the i-th element of vector $\boldsymbol{w}_j$, and b_j represents the regression weight for the j-th latent variable, $b_j = \boldsymbol{u}_j^T \boldsymbol{t}_j$.

We then extend the definition of VIP by introducing block importance score (BIS). The BIS ranks the predictive power of each block in the PLS model. The BIS on a multi-level blocks exhibits a "hierarchical" modeling, which first tries to find the important blocks from the multi-level/multi-size blocks and retrieve more detail information from the blocks which have better importance score. Algorithm 2 and Fig. 4 show the procedure of creating the BIFM-PLS model.

The BIFM-PLS algorithm consists of two important stages: building the block importance and projecting the features on the important block. Let n be the number of blocks for each HOG and LBP features, m be the total number of concatenated HOG-LBP features in one block, $\boldsymbol{f}_i$ denotes the feature sets at the i-th block, and $\boldsymbol{f}_{ij}$ represents the j-th feature at the i-th block with $i = \{1, \ldots, n\}$ and $j = \{1, \ldots, m\}$.

In the first stage, we extract a PLS model, take first p_1 latent variables for each block, and concatenate them to build a model $\boldsymbol{f}^{first}$, as follows

$$\boldsymbol{f}_i^{first} = PLS(\boldsymbol{f}_i, p_1), \tag{6}$$

where $PLS(\boldsymbol{f}_i, p_1)$ is a function for extracting p_1 elements from $\boldsymbol{f}_i$. Here $\boldsymbol{f}_i^{first}$ has element $\boldsymbol{f}_{i,j_1}^{first}$ where $j_1 = \{1, \ldots, p_1\}$.

We assume that a small number of p_1 are enough to see the contribution of each block to the orientation estimation, since the PLS considers the response variables (i.e. class labels). To see the importance of each block, we compute the VIP scores using Eq. (5) and get the block importance score (BIS) as

$$BIS_i = \frac{1}{p_1} \sum_{j_1=1}^{p_1} VIP\left(\boldsymbol{f}_{i,j_1}^{first}\right), \qquad i = 1, \ldots, n. \tag{7}$$

[1] According to the original paper [2], the uniform patterns contain at most two bit transitions from 0 to 1 and vice versa. For an 8-bit data, there are 58 uniform patterns, and the other patterns which have more than two bit transitions are grouped into one label, so we have the total 59 labels.

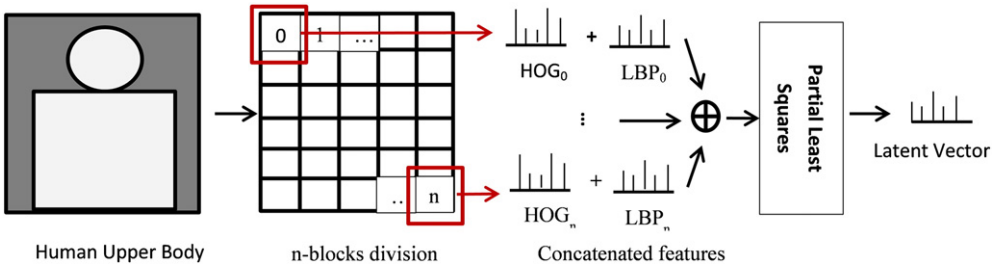

Fig. 3. Extracting features using CFM-PLS.

The top part of Fig. 4 shows the processes explained by Eqs. (6) and (7) above.

In the second stage, we build the BIFM feature vectors by calculating and concatenating the first p_2 latent variables on the important blocks, similar to Eq. (6)

$$\begin{aligned} \boldsymbol{f}_i^{second} &= PLS(\boldsymbol{f}_i, p_2), \\ \boldsymbol{f}^{BIFM} &= \bigcup_i \boldsymbol{f}_i^{second}, \qquad \{\forall i | BIS_i > \eth\}. \end{aligned} \tag{8}$$

The blocks which have BIS under the threshold $\eth$, which mean less important blocks, are then discarded. These procedures are illustrated by the bottom part of Fig. 4.

Here we use $p_2 > p_1$, to get more detail information on each important block. We will examine the proper value of p_1 and p_2 in the experimental section (Section 4.4).

Algorithm 2. BIFM-PLS algorithm

Require:
n: number of blocks
p_1: number of latent vectors at stage 1
p_2: number of latent vectors at stage 2

Ensure:
$\boldsymbol{f}_{BIFM}$: the reduced set of features vectors

procedure CreateBIFM()
  BIFM_First(p_1)
  $\boldsymbol{f}_{BIFM} \leftarrow$ BIFM_Reproject(p_2)
end procedure

procedure BIFM_First(p_1)
  for $i = 1 \rightarrow n$ **do**
    $\boldsymbol{f}_i^{first} \leftarrow \text{PLS}(\boldsymbol{f}_i, p_1)$
  end for
end procedure

function BIFM_Reproject(p_2)
  for $i = 1 \rightarrow n$ **do**
    if $\text{BIS}(\boldsymbol{f}_i^{first}) <$ threshold **then**
      $\boldsymbol{f}_{BIFM} \leftarrow \bigcup \text{PLS}(\boldsymbol{f}_i, p_2)$
    end if
  end for
  return $\boldsymbol{f}_{BIFM}$
end function

2.5. Random Forest

One of the notable classifier which works well on the multi-class data is Random Forest, introduced by Breiman [8]. It is an ensemble learning method which combines the prediction of many decision trees using a majority vote mechanism. The Random Forest is devoted for its accuracy on the large dataset and multi-class learning. These advantages make us choose the Random Forest for training our eight-orientation classification problem with a large set of features.

Our Random Forest is constructed by multiple trees $T = \{T_1, T_2, \ldots, T_N\}$, where N is number of trees. Let $\{d_i \in \mathcal{D}\}_{i=1\ldots K}$ denote K training sets and $\{c_i \in \mathcal{C}\}_{i=1\ldots K}$ be its corresponding labels or classes (in our case, we have eight classes, $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_8\}$), where $\mathcal{D} \subset \mathbb{R}^M$ is the feature space. In the training phase, the Random Forest learns the classification function $T : \mathcal{D} \rightarrow \mathcal{C}$. Details of the Random Forest algorithm can be discovered at the original paper [8].We use a linear split function [19]

$$\Phi(\boldsymbol{f}) = \boldsymbol{q}^T \boldsymbol{f} + \boldsymbol{z}, \tag{9}$$

where q is a vector which has the same dimensions as the feature vector $\boldsymbol{f}$ and $\boldsymbol{z}$ is a random number. The recursive training is run until the stopping criteria are reached, i.e. the maximum depth is met or no further information gain can be drawn.

In the testing phase, for a test case d, the Random Forest provides the posterior probability of each class as

$$p(c|d) = \frac{1}{N} \sum_{i=1}^{N} p_i(c|d), \tag{10}$$

where

$$p_i(c|d) = \frac{\Lambda_{i,c}}{\sum_{j=c_1}^{c_8} \Lambda_{i,j}}, \tag{11}$$

$p_i(c|d)$ is the probability estimation of class $c \in \mathcal{C}$ given by the i^{th} tree, and $\Lambda_{i,c}$ is the number of leaves in the i^{th} tree which votes for class c. The overall multi-class decision function of the forest is then defined as

$$C(d) = \arg\max_{c \in \mathcal{C}} p(c|d). \tag{12}$$

3. Integration of orientation estimation and tracking

In the normal situation, the possibility that the person body orientation will be the same with its movements increases along with its speed. Based on this observation, our orientation estimation system is built using an assumption that the human body orientation is aligned with the direction of the human movements. In this case, we utilize both the result of the orientation detections and that of human movement

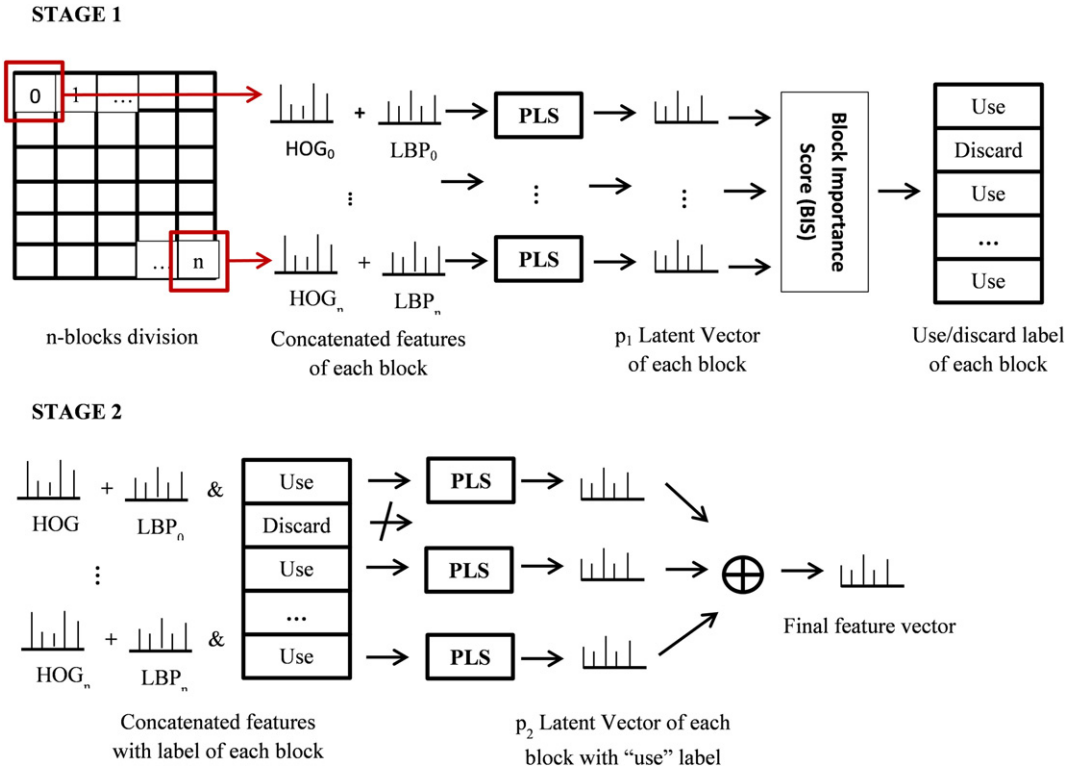

Fig. 4. Extracting features using BIFM-PLS.

estimation. On the opposite, we only rely on the orientation information from detections when the human movements are slow or even in a static condition.

To handle those matters, we predict the movement of persons in the world coordinate. We then use the UKF framework to combine the orientation estimation from the detection results and the movement predictions. The idea of coupling the detection result and the person movement is also used by [5], but at least three (3) things distinct our works from the [5]; the baseline (upper body vs whole body), features (HOG–LBP–PLS vs Sparse-HOG), and the framework (UKF vs particle filter).

3.1. Estimating human movement through its position in the real world

We derive movement of the persons from the change of their positions. We follow the work of [9] and [10] for projecting the position of each detected person from the 2D image coordinate to the 3D world coordinate. Let us consider a pinhole camera model, with the following parameters: focal length f_c, camera height y_c, horizontal center point μ_c, and horizon position ν_0. According to [10], the projection to the world coordinate is given by

$$W_d = \begin{bmatrix} \frac{y_c(\mu_d - \mu_c)}{\nu_d - \nu_0} \\ \frac{f_c y_c}{\nu_d - \nu_0} \\ \frac{h_d y_c}{\nu_d - \nu_0} \end{bmatrix}, \tag{13}$$

where (μ_d, ν_d) is the bottom center point of the extended bounding box,[2] and h_d is the height of each detected person d in the 2D image. Vector $W_d = [x_d^{world}, y_d^{world}, h_d^{world}]^T$ denotes the position $(x_d^{world}, y_d^{world})$ in the real world relative to the camera position and the height h_d^{world} of the detected person d in the image.

The horizon position v_0 is obtained by collecting line segments in the image using Hough line detector and running RANSAC to evaluate all segments and get the intersection. This horizon estimation is done off-line[3] and we use it as a pre-calibrated value for the on-line tracking. For each frame, we send the position $(x_d^{world}, y_d^{world})$ to the tracker for getting the movement estimation in the real world.

3.2. Tracking strategies

3.2.1. State and observation models

Our state model is decomposed from the person position (x_k, y_k), its derivative $(\dot{x}_k, \dot{y}_k)$, and the orientation components $(\varphi_k^M, \varphi_k^D, \theta_k)$ where φ_k^M and φ_k^D respectively denote the person orientation due to the movement and the detection estimation and θ_k denotes the final orientation of the person after considering the movement and the detection result, as stated by following tuple, $\mathcal{F}(\mathcal{X}) = \{x_k, y_k, \dot{x}_k, \dot{y}_k, \varphi_k^M, \varphi_k^D, \theta_k\}$. A constant velocity model is used for representing the person position and its derivative, as follows

$$\begin{cases} x_k &= x_{k-1} + \dot{x}_{k-1}\delta t + \epsilon_x, \\ y_k &= y_{k-1} + \dot{y}_{k-1}\delta t + \epsilon_y, \\ \dot{x}_k &= \dot{x}_{k-1} + \epsilon_{\dot{x}}, \\ \dot{y}_k &= \dot{y}_{k-1} + \epsilon_{\dot{y}}, \end{cases} \tag{14}$$

where $\{\epsilon_x, \epsilon_y, \epsilon_{\dot{x}}, \epsilon_{\dot{y}}\}$ are the gaussian noises for each component, and δ_t is the time sampling.

[2] The extended bounding box is calculated by scaling the height of the bounding box to that of the human body, so that the bottom center is on the ground plane.

[3] To keep the speed of algorithm, we compute and use the horizon estimate as a constant. We assume the camera tilt is small (or in other words, the ground plane where the robot runs is flat).

The orientation components of $\mathcal{F}(\mathcal{X})$ are then described as

$$\begin{cases} \varphi_k^M &= arctan\left(\frac{\dot{y}_{k-1}}{\dot{x}_{k-1}}\right) + \epsilon_{\varphi_M}(v_{k-1}), \\ \varphi_k^D &= \varphi_{k-1}^D + \epsilon_{\varphi_D}, \\ \theta_k &= \varphi_{k-1}^D + \frac{\omega\left(1-e^{-v_{k-1}}\right)}{1+e^{-v_{k-1}}}\left(\varphi_{k-1}^M - \varphi_{k-1}^D\right) + \epsilon_\theta, \end{cases} \tag{15}$$

where $\left\{\epsilon_{\varphi_D}, \epsilon_\theta\right\}$ are the gaussian noises and ω is a constant.

The noise function $\epsilon_{\varphi_M}(v_{k-1})$ in the first line of Eq. (15) is defined as

$$\begin{aligned} \epsilon_{\varphi_M}(v_{k-1}) &= \mathcal{N}(0, g(v_{k-1})), \\ g(v_{k-1}) &= \sigma_v exp^{-v_{k-1}}, \\ v_{k-1} &= \sqrt{\dot{x}_{k-1}^2 + \dot{y}_{k-1}^2}, \end{aligned} \tag{16}$$

where v_{k-1} is the magnitude of the person movement, and σ_v is a constant. Using Eq. (16), the noise function $\epsilon_{\varphi_M}(v_{k-1})$ can be read as a zero-mean gaussian of which the variance varies *w.r.t* the person movement v.

Eq. (15) suggests the influence of the person movement to the orientation estimation. This equation, together with Eq. (16), tells us that when the velocity of a person is small enough ($v_{k-1} \approx 0$), the uncertainty of the movement orientation becomes large (see $\epsilon_{\varphi_M}(v_{k-1})$), and the orientation estimation will mainly depend on φ_k^D (which is updated using the orientation detection result in the observation model (see Eq. (17))). On the contrary, when the velocity is large then the movement orientation becomes more reliable and the orientation estimation depends on both detection and the person movement.

Here the constant ω has a duty for controlling the influence of the detection and the person movement to the overall orientation. The value of ω is later investigated in the experimental section (Section 4.5).

We then use the observation model for the person position in the world coordinate (derived from Eq. (13)) and for the person orientation from the result of Section 2, as follows

$$\mathcal{H}(\mathcal{X}) = \begin{cases} \mu_k &= \frac{x_k\left(\frac{f_c y_c}{y_k}\right)}{y_c} + \mu_c + \epsilon_\mu \\ \nu_k &= \frac{f_c y_c}{y_k} + \nu_0 + \epsilon_\nu \\ h_k &= \frac{\frac{f_c y_c}{y_k} h_d}{y_c} + \epsilon_h \\ C_k &= f\left(C(d), \varphi_k^D\right) + \epsilon_c \end{cases} \tag{17}$$

where (x_k, y_k) denote the person position, (f_c, y_c, μ_c, ν_0) are the camera parameters (focal length f_c, camera height y_c, horizontal center point μ_c, and horizon position ν_0), and $\{\epsilon_\mu, \epsilon_\nu, \epsilon_h, \epsilon_c\}$ are the gaussian noises for each observation component. μ_k, ν_k, and h_k have the same definition with μ_d, ν_d, and h_d in Section 3.1. The function $f(C(d), \varphi_k^D)$ maps the result of the multi-class decision function $C(d)$ (Eq. (12)) into their equivalent angle of the orientation φ_k^D.

3.2.2. Unscented Kalman Filter tracker

For choosing the tracker, we consider our hardware limitation and the system nonlinearities. Several well-known filters can be adopted for handling the nonlinearities, such as Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and particle filter. Here we choose UKF because we want to avoid the resource costs of the particle filter (used by [5]) and the calculus of the Jacobian matrices used in the EKF.

The UKF [20] employs the unscented transform, a deterministic sampling technique, to take a minimal set of sample points called sigma points around the mean. Consider the current state with the mean $\hat{x}$ and its covariance P_x. We build a set of 2 L + 1 sigma points matrix $\mathcal{X}$, as follows

$$\begin{aligned} \mathcal{X}_0 &= x \\ \mathcal{X}_{i,k-1} &= \hat{x} + \left(\sqrt{(L+\lambda)P_x}\right)_i, i = \{1, \ldots, L\} \\ \mathcal{X}_{i,k-1} &= \hat{x} - \left(\sqrt{(L+\lambda)P_x}\right)_{i-L}, i = \{L+1, \ldots, 2L\} \\ \lambda &= \alpha^2(L+\kappa) - L, \end{aligned} \tag{18}$$

where L is the dimension of the augmented state, α and κ are constants for controlling the spread of the sigma points.

We then define $W^{(m)}$ and $W^{(c)}$ as the weight for the mean and covariance respectively, given by

$$\begin{aligned} \boldsymbol{W}_0^{(m)} &= \frac{\lambda}{(L+\lambda)} \\ \boldsymbol{W}_0^{(c)} &= \frac{\lambda}{(L+\lambda)} + \left(1 - \alpha^2 + \beta\right) \\ \boldsymbol{W}_i^{(m)} &= W_i^{(c)} = \frac{1}{2(L+\lambda)}, i = \{1, \ldots, 2L\} \end{aligned} \tag{19}$$

where β integrates prior knowledge of the distribution of $\hat{x}$. We use the default value, $\alpha = 10^{-3}$, $\beta = 2$, and $\kappa = 0$.

We compute mean and covariance of the prior estimation ($\mathcal{X}_{k|k-1}$ and P_k^-) using the state model $\mathcal{F}(\mathcal{X})$ in Eqs. (14) and (15), and the sigma points above as follows

$$\begin{aligned} \mathcal{X}_{i,k|k-1} &= \mathcal{F}\left(\mathcal{X}_{i,k-1}\right), i = \{1, \ldots, 2L\} \\ \hat{x}_k^- &= \sum_{i=0}^{2L} \boldsymbol{W}_i^{(m)} \mathcal{X}_{i,k|k-1} \\ \boldsymbol{P}_k^- &= \sum_{i=0}^{2L} \boldsymbol{W}_i^{(c)} \left[\mathcal{X}_{i,k|k-1} - \hat{x}_k^-\right]\left[\mathcal{X}_{i,k|k-1} - \hat{x}_k^-\right]^T + Q \end{aligned} \tag{20}$$

where Q is the covariance of the process noises.

In the measurement update phase, we project the sigma points through the observation function $\mathcal{H}(X)$

$$\begin{aligned} Z_{i,k|k-1} &= \mathcal{H}\left(X_{i,k|k-1}\right), i = \{1, \ldots, 2L\} \\ \hat{z}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} Z_{i,k|k-1}, \\ P_{z_k z_k} &= \sum_{i=0}^{2L} W_i^{(c)} \left[Z_{i,k|k-1} - \hat{z}_k^-\right]\left[Z_{i,k|k-1} - \hat{z}_k^-\right]^T + R, \end{aligned} \tag{21}$$

where $\hat{z}_k^-$ denotes the predicted measurement, $P_{z_k z_k}$ is its covariance, and R is the covariance of the observation noises. The state measurement cross-covariance $P_{x_k z_k}$ and the Kalman gain K are computed as

$$\begin{aligned} P_{x_k z_k} &= \sum_{i=0}^{2L} W_i^{(c)} \left[X_{i,k|k-1} - \hat{x}_k^-\right]\left[Z_{i,k|k-1} - \hat{z}_k^-\right]^T, \\ K &= P_{x_k z_k} P_{z_k z_k}^{-1}. \end{aligned} \tag{22}$$

The mean and covariance of the posterior estimation are then calculated as

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + \boldsymbol{K}(z_k - \hat{z}_k^-), \\ \boldsymbol{P}_k &= \boldsymbol{P}_k^- - \boldsymbol{K}\boldsymbol{P}_{z_k z_k}\boldsymbol{K}^T. \end{aligned} \tag{23}$$

We also give to the tracker, the color histogram information H_{tr} retrieved from the lower third of the bounding box, from which we expect to get clothing features.[4] In the implementation using a moving robot

[4] In the upper body bounding box, the lower third part usually contains the human shoulder and body. We can expect that the cloth color can be retrieved from this region and does not change much during the tracking process.

Fig. 5. Region of Interest (ROI) for tracking. The red transparent shadows are places which has a high probability that person may appear to the scene from the side. The green transparent shadow is the ROI of the tracker. The blue cylinder denotes the current tracker bounding box.

(see Section 4.7), the camera movement is currently compensated by using the odometry of the robot. The use of Kanade–Lucas–Tomasi (KLT) features tracker [10] or the visual odometry [11] for compensating the camera movement are further investigated in the future.

3.2.3. Association

We treat each detection of the human upper body as an observation, to be associated with the trackers. For every observation, we use the position and histogram information for calculating the relative distance to each tracker $\Delta_{ob,tr}$, given by

$$\Delta^{P}_{ob,tr} = \frac{e^{-\frac{1}{2}(\overline{\mu}_{ob}-\overline{\mu}_{tr})^{T}\left(\sum_{ob}+\sum_{tr}\right)^{-1}(\overline{\mu}_{ob}-\overline{\mu}_{tr})}}{\sqrt{2\phi\left|\left(\sum_{ob}+\sum_{tr}\right)\right|}} \tag{24}$$

$$\Delta^{H}_{ob,tr} = \frac{\sum_{I}\left(H_{ob}(I)-\overline{H}_{ob}\right)\left(H_{tr}(I)-\overline{H}_{tr}\right)}{\sqrt{\sum_{I}\left(H_{ob}(I)-\overline{H}_{ob}\right)^{2}\sum_{I}\left(H_{tr}(I)-\overline{H}_{tr}\right)^{2}}} \tag{25}$$

$$\Delta_{ob,tr} = \eta\Delta^{P}_{ob,tr} + \varsigma\Delta^{H}_{ob,tr} \tag{26}$$

where $\Delta^{P}_{ob,tr}$ is the gaussian correlation between mean and covariance of the observed position $(\overline{\mu}_{ob}, \sum_{ob})$ and the tracker position $(\overline{\mu}_{tr}, \sum_{tr})$. $\Delta^{H}_{ob,tr}$ is the color histogram correlation distance of the observation H_{ob} and the tracker H_{tr}, and η and ς are constants.

Under the nearest neighbor assumption, each observation is assigned to the tracker when the distance $\Delta_{ob,tr}$ is below the threshold. A new tracker is born from the observation which has no association, and any track which is not associated with any observation and has a large uncertainty is then deleted.

3.2.4. ROI-based tracking

For reducing the calculation time, we do a hierarchical Region of Interest (ROI)-based tracking (including the orientation estimation) along the frame sequences. First, we search the whole space of the image to obtain initial observations and trackers. The next sequences utilize the position information from the trackers as the ROI. We do the detection around the ROI and in the area where persons may come up to the scene (see Fig. 5). This technique[5] considerably reduces the time for detection. Lastly, every 15 frames, we do the whole space search to anticipate the missing detection.

4. Experiments

Size of all images and camera sequences used in our experiments is 640 × 480. All of experiments were done using C++ implementation on a laptop PC (Core2Duo 2.1 GHz, 2 GB memory, and Windows 7 OS).

The evaluation of our method starts from explaining the dataset used by our system. We then build an analysis to support the advantages of using our method for the human orientation estimation, by comparing it with several methods. We also evaluate performance of the orientation estimation and tracking integration. Lastly, we implement our system to the real environment using a camera attached on a mobile robot.

4.1. Dataset

First, we create our upper body dataset by cropping the INRIA [1] and Fudan-Penn [6] data into 48 × 64 pixel images containing the upper-half body of persons. We also add the CALVIN upper body dataset [7] into our dataset, so that we have 4250 positive samples of the human upper body. Around 3000 positive samples are used for training the upper body detector, and the rest is for testing purpose. 2500 negative samples are created from images which do not contain the human upper body, including the bottom part of the human body. From now, we refer it as *dataset A*.[6]

To do a comprehensive test of the human upper body orientation estimation, we use several datasets for both static and dynamic scenes. For the static scenes, we use TUD-Multiview dataset [3] which is also used by the other state-of-the-art papers ([5,21]). This dataset consists of 1486 images for training, 248 images for validation, and 248 images for testing. The TUD-Multiview dataset is annotated with bounding boxes of full body and eight orientation classes. For our purpose, we change the bounding boxes into the half upper part of the body.

We also use the dataset created for upper body training (*dataset A*), for the orientation classification purpose. We then separate the training samples of the above dataset into eight classes representing the eight orientation of the human body (see Fig. 1). The testing samples are treated in the same manner with the training samples. This dataset is then called as *dataset B*.

For the dynamic scenes, we use TUD-Stadtmitte dataset [3] which contains 200 frames of the street scenes with several pedestrians crossing the street with a complex environment and many occlusions, and the camera calibration data.[7] Here we use the raw video for the TUD-Stadtmitte dataset without annotations.

We then take an indoor video of our laboratory (from here, it is referred as *InLab dataset*). This video contains 487 frames with the number of persons varies from zero to three persons on each frame, and also the camera calibration data.

We summarize the usage of each dataset, as follows:

- *Dataset A* is used for training and testing the human upper body detection;
- *Dataset B* is used for evaluating the orientation estimation under various features and classifiers (see Sections 4.3 and 4.4);
- TUD-Multiview dataset is used for comparison with the state-of-the-arts (see Section 4.6);

[5] Basically, this heuristic method is enough for our cases. For more general scenarios, we open this problem to the interested reader.

[6] This generic naming is for simplicity only, not intended to give a new name to the existing databases.

[7] Camera calibration data of the TUD-Stadtmitte dataset is provided at *http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html*.

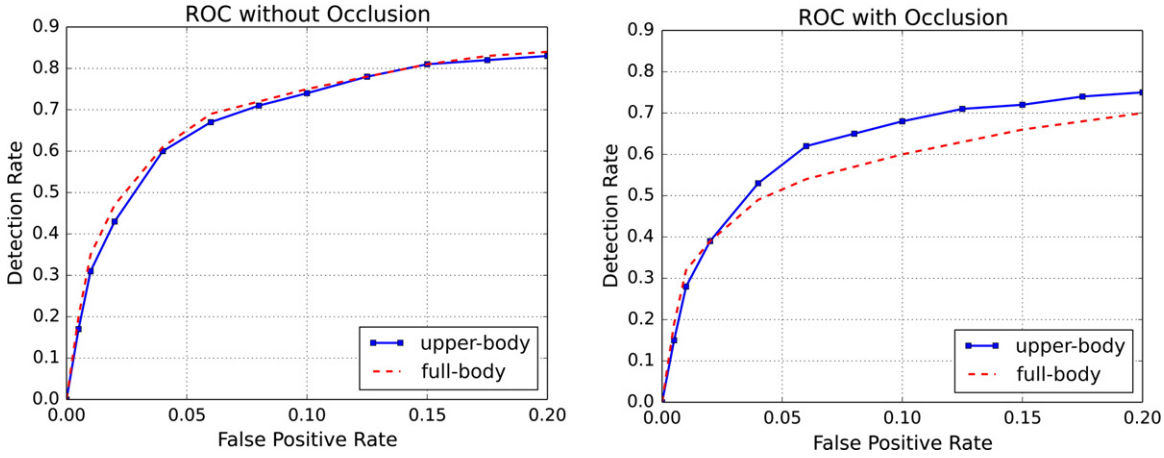

Fig. 6. Performance of the human full body and upper body detection. The left graph shows the result on the dataset with no occlusion. The right graph shows the result on the occluded dataset.

- Combination of *subset B* and TUD-Multiview datasets are then used for training the orientation model to be used in the dynamic scenes, i.e. TUD-Stadtmitte and InLab datasets (see Section 4.5), and the real world application (see Section 4.7).

4.2. Human upper body detection performance

At the beginning, we want to show the reason for choosing the human upper body over the full body detection in our system. We build the human full body dataset from a subset of the INRIA dataset [1], and for the upper body dataset, we use the *dataset A* mentioned in Section 4.1. We then create two testing samples; the first samples contain a subset of testing samples of the INRIA which do not contain occluded person, and in the second ones we add several images with persons occluded by the chairs, tables, and furniture in additional to the testing samples of the INRIA. The training of both cases is then performed using Boost-Cascade method, as mentioned in Section 2.2. The ROC of both occluded and non-occluded cases are shown in Fig. 6.

The results show that the human upper body detector works better on the occluded dataset. This result suggests the use of the human upper body detection on the real cases such as an indoor environment with many tables, chairs, and furniture, rather than using the full body model.

Table 1
Evaluation of the orientation estimation using various features and classifiers.

Feature	Classifier	Accuracy	
		TUD-Multiview	Dataset B
HOG	MultiSVM	0.35	0.34
HOG	MultiBoost	0.37	0.38
HOG	Random forest	0.44	0.46
HOG + LBP	MultiSVM	0.45	0.43
HOG + LBP	MultiBoost	0.48	0.45
HOG + LBP	Random forest	0.53	0.52
HOG + LBP + CFM-PLS	MultiSVM	0.45	0.42
HOG + LBP + CFM-PLS	MultiBoost	0.50	0.47
HOG + LBP + CFM-PLS	Random forest	0.54	0.54
HOG + LBP + BIFM-PLS	MultiSVM	0.60	0.57
HOG + LBP + BIFM-PLS	MultiBoost	0.60	0.58
HOG + LBP + BIFM-PLS	Random forest	0.64	0.60

4.3. Evaluation of the orientation estimation using various features and classifiers

First, we conduct experiments to see the influence of the features, the models, and the classifiers to the orientation estimation results by using the *dataset B*. Following parameters are used for the experiments; we use the multi-level HOG and LBP features explained in the beginning of this paper; for the multi-class SVM classifier [13], we use a standard RBF kernel with gamma set to 4e-4, and regularization parameter is set to 1.0; for the MultiBoost [14], we use "FilterBoost" for the strong learner and "SingleStump" for the base learner; for the Random Forest, the number of trees is set to 100 and the maximum depth of trees is set to 25; the latent vector for CFM-PLS is set to 15, and for the BIFM-PLS, we set p_1 to 3 and p_2 to 15 (we will discuss these values of the PLS models in the next subsection).

Table 1 shows the experimental result using the combination of features and classifiers. Based on this table, the combination of multi-level HOG-LBP features, BIFM-PLS model, and Random Forest classifier, is

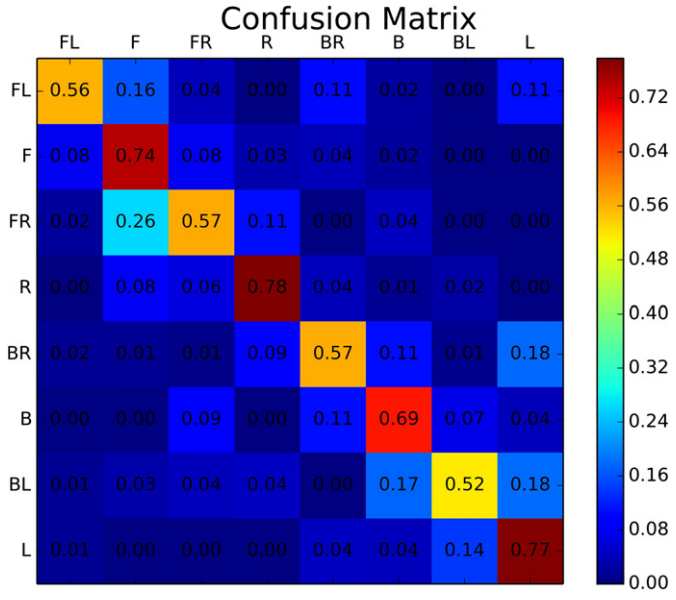

Fig. 7. Confusion matrix on TUD-Multiview dataset using BIFM-PLS and random forest.

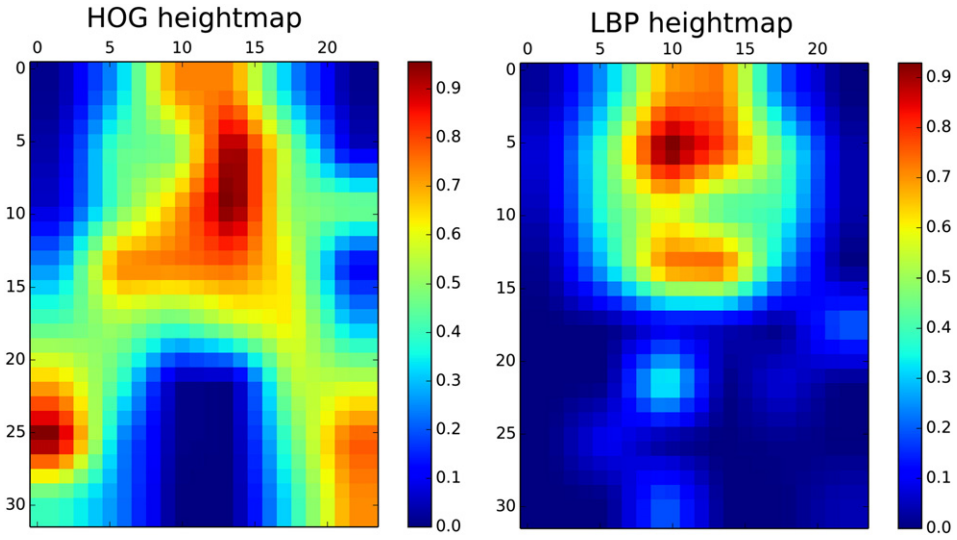

Fig. 8. The block importance of each feature using the weight of the first projection vector. The left figure shows the importance of one level block due to the HOG features. The right figure shows the importance of one level block due to the LBP features. Red color represents the high importance.

superior against the others. In general, the combination of HOG and LBP features is better than using only HOG features. The RF classifier also gives better impact than using MultiSVM and MultiBoost. The interesting part is that the usage of PLS models yields better result of the orientation estimation than the concatenated features. We will discuss these matters in the next subsection.

Fig. 7 shows the distribution of the orientation estimation. From this figure, we can conclude that estimating an oblique direction is more difficult than detecting the perpendicular one.

4.4. Analysis of the PLS models

We now discuss about how our PLS models (BIFM-PLS) give a significant contribution to the orientation estimation. We extract the block importance score (BIS) of each LBP and HOG features using the BIFM-PLS algorithm, and draw the weight of the first projection vector.

Fig. 8 shows the contribution of each features to the orientation estimation. The left figure, which utilizes the HOG features, shows that the areas around the edges of the body have a high importance, and the background tends to have a low importance. It means that the HOG extracts the shape of the body for the orientation estimation. In the right figure, we can see that the high importance area is around the head area and the body, but not for the background and the clothing. Here we can understand that the LBP captures the head features for the orientation estimation, while the clothing and background which vary from one sample to the others are discarded.

Based on Fig. 8, and supported by the results in Table 1, we show the effect of the BIFM-PLS model to the classification. Even the classifier such as the Random Forest is noted to be able to extract the importance of the features (for example, in [22]), here the BIFM-PLS removes the "noisy" areas (such as the various background and clothing) as shown in Fig. 8, and helps the classifier focus to do the classification on the high importance features.

Table 2
Performance of PLS and PCA for the orientation estimation.

Method	Accuracy	
	TUD Multiview	Dataset B
HOG + LBP + CFM-PLS + RF	0.54	0.54
HOG + LBP + BIFM-PLS + RF	0.64	0.60
HOG + LBP + PCA + RF	0.52	0.51

We can also see the advantages of the PLS models as a dimensional reduction algorithm. We use another popular dimensional reduction algorithm, Principal Component Analysis (PCA), as the baseline. Table 2 shows the superiority of the PLS models against the PCA. It is understandable because unlike the PCA, the PLS also considers the class labels besides the variance of the samples.

We then discuss about the effect of the constants p_1 and p_2 used in the BIFM-PLS algorithm, shown in Fig. 9. p_1 represents how well a block contributes to the orientation estimation, and p_2 examines the contribution of a feature inside a block. By examining Fig. 9, we choose the optimal value for both constants, $p_1 = 3$ and $p_2 = 15$. Over those values, we can consider it as the data overfit.

4.5. Evaluation on integrated orientation estimation and tracking performances

The next experiments are intended for evaluating the performance of the integrated orientation estimation and tracking. Here we use the TUD-Stadtmitte and InLab datasets for the evaluations. We also investigate the influence of choosing various values of ω as mentioned in Eq. (15). The higher value of ω means that the object movement will give a higher influence to the orientation estimation result.

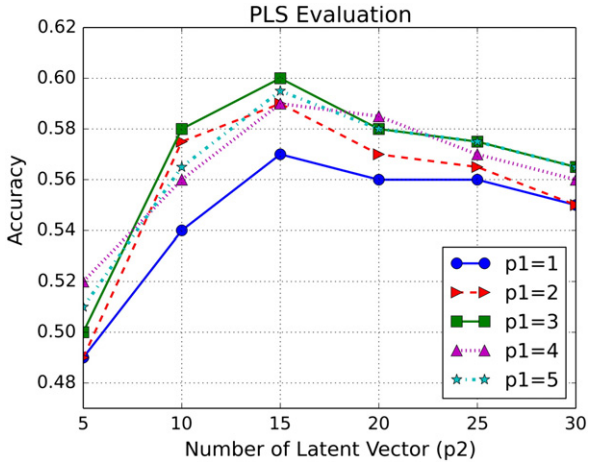

Fig. 9. The effect of varying p_1 and p_2 value of the BIFM-PLS to the orientation estimation results.

Fig. 10. Human upper body orientation estimation results on TUD-Stadtmitte datasets using our proposed framework. Top sequence shows the detection (bounding boxes) and orientation estimation (arrows) results. Bottom sequence shows the estimated position of each detected person (circles) in the 3D world coordinate, with respect to the person position in the top images. Two orange line segments in the bottom sequences denote the camera FOV.

Fig. 10 shows the performance of our proposed system using the TUD-Stadtmitte dataset. The top sequence shows the orientation detections and tracking results. The bottom sequence shows the person locations and movement predictions in the 3D world coordinate, relative to the position and FOV of the camera.

We then test our framework on InLab dataset, as shown in Fig. 11 (please refer to Fig. 10 for figure property explanation). Fig. 11 exhibits the robustness of the multi target tracking and orientation estimation under many occlusions.

Fig. 12 shows the effect of changing ω value to the overall orientation estimation results. We change the value of ω from 0, which means that the orientation estimation depends only on the detection, to $\omega = 1$ which represents a heavy dependency to the object movements.

In the TUD-Stadtmitte dataset, the orientation estimation based on the object movements ($\omega = 1$) is relatively high compared to the one based on the detection ($\omega = 0$), due to the consistent movement of each person inside the video sequences, which gives a high confidence to the movement estimation. Contrary, the orientation estimation based on the object movement in the InLab dataset is lower because the persons frequently change their direction. Overall, combining both detection and movement estimation tends to make a higher orientation estimation results rather than solely depends on the detection or the movement estimation, and in our cases we choose $\omega = 0.5$.

4.6. Comparison with the state-of-the-art

We compare our algorithm to the state-of-the-art papers such as [3,5], and [22] using the TUD-Multiview dataset. Since the state-of-the-art papers use the whole body information for estimating the orientation, we cannot directly use the result of their papers as the comparison. Instead, we need to test their methods using the upper body information. Unfortunately, the authors of those papers do not provide any implementation code.

To overcome the problem above, we re-implement their algorithms based on our understanding to their papers. We then evaluate it using the full body information. We expect that the result will be similar with the one mentioned in their respective paper. The first and second columns of Table 3 show the comparison of the original and the re-implemented version of the state-of-the-art papers. We can see that our re-implementation gives a close result to the one mentioned in each paper. Until this stage, our algorithm beats all of Andriluka's and Chen's works and comparable to Baltieri's work, even though they use the full body information. Now we can assume that our re-implementation of the state-of-the-art can be used for further comparison.

We then use the re-implementation of the state-of-the-art for making a fair comparison with our algorithm, i.e. by applying the

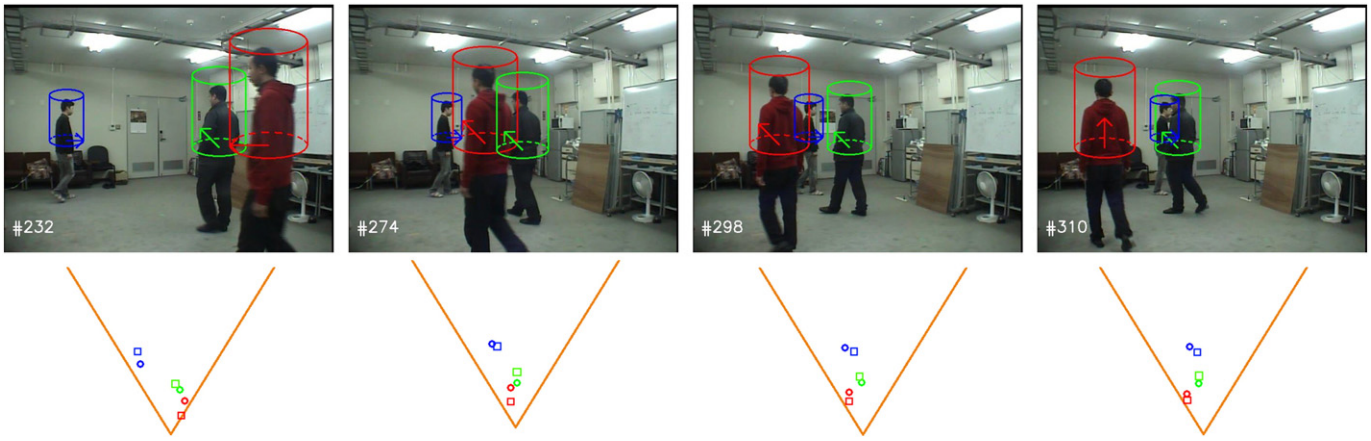

Fig. 11. Human upper body orientation estimation results in an indoor environment. Top sequence shows the detection (bounding boxes) and orientation estimation (arrows) results. Bottom sequence shows the position of estimated each detected person (circles) and the ground truth (rectangles) in the 3D world coordinate, with respect to the person position in the top images. Two orange line segments in the bottom sequences denote the camera FOV.

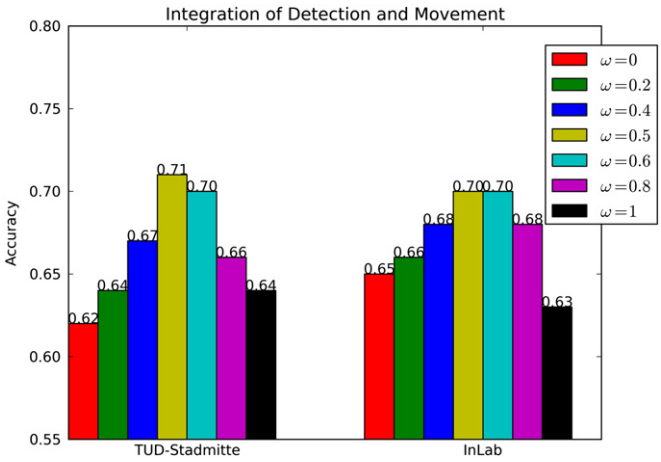

Fig. 12. The effect of varying ω value to the integration of orientation results.

Table 3
Comparison of the state-of-the-art algorithms.

Method	Accuracy full body		Accuracy
	Paper	Our implementation	Upper body
Andriluka [3] — Max	0.31	0.29	0.20
Andriluka [3] — SVM	0.42	0.39	0.35
Andriluka [3] — SVM_adj	0.35	0.33	0.27
Chen [5] — Sparse	0.55	0.52	0.40
Baltieri [21] — AWG	0.65	0.59	0.51
Ours	N/A	N/A	0.64

same upper body dataset which contains less information. Each algorithm is then trained and tested using the upper body version of the TUD-Multiview dataset. The last column of Table 3 shows the result of each algorithm using only the upper body information. The result of each state-of-the-art degrades significantly, as less information is available for obtaining the body orientation. It also shows the importance of using various cues. Other works use only the shape features which decrease the performance when the information becomes less. On the other hand, our algorithm uses the shape and texture features simultaneously to overcome those problems and gains the best performance on the upper body orientation estimation.

4.7. Orientation estimation on moving camera

In this experiment, we attach a monocular camera on the mobile robot base. The robot is then controlled to move while performing the human upper body orientation estimation and tracking in real-time. The experiment was performed at the university cafeteria, with the total of 190 frames. Fig. 13 shows the experimental results using the camera with a moving base. We can achieve the accuracy rate of 0.70 and the frame rate of 5–12 Hz, fast enough to be used for an on-line purpose. Once again, it shows the consistency and robustness of our orientation estimation and tracking system.

5. Conclusions

We have described a framework of orientation estimation and tracking. Our human upper body orientation classification system, utilizing a partial least squares model-based shape-texture features combined with the random forest, is proved to work better than any existing methods. Its integration with the tracking system boosts up the performance of the orientation estimation even further. Another notable result is that our system works real-time, giving a possibility to be used in the real robot application such as the person tracking.

Future work for our system, besides the implementation on a real robot for specific purposes, is to combine it with other sensors such as laser range finders. By adopting multi-sensory fusion, we expect to build a more robust system for person localization. There are also possibilities for choosing and adding better heuristics in the integration of orientation estimation and tracking, such as a better way for selecting the ROI, so that the system will be applicable for more general scenarios.

References

[1] N. Dalal, B. Briggs, Histograms of oriented gradients for human detection, IEEE Conf. Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[2] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 24, 7, 2002, pp. 971–987.
[3] M. Andriluka, S. Roth, B. Schiele, Monocular 3D pose estimation and tracking by detection, IEEE Conf. on Computer Vision and Pattern Recognition, 2010, pp. 623–630.
[4] C. Weinrich, C. Vollmer, H. Gross, Estimation of human upper body orientation for mobile robotics using an SVM decision tree on monocular images, Int. Conf. on Intelligent Robots and Systems, 2012, pp. 2147–2152.
[5] C. Chen, A. Heili, J. Odobez, Combined estimation of location and body pose in surveillance video, IEEE Conf. on Advanced Video and Signal-Based Surveillance, 2011, pp. 5–10.

Fig. 13. Human upper body orientation estimation results in the cafetaria with a moving camera.

[6] L. Wang, J. Shi, G. Song, I. Shen, Object detection combining recognition and segmentation, Asian Conference on Computer Vision, 2007, pp. 189–199.
[7] V. Ferrari, M.J. Marn-Jiminez, A. Zisserman, Progressive search space reduction for human pose estimation, IEEE Conf. on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[8] L. Breiman, Random Forest, Mach. Learn. 45 (2001) 5–32.
[9] D. Hoiem, A.A. Efros, M. Hebert, Putting objects in perspective, IEEE Conf. on Computer Vision and Pattern Recognition, 2006, pp. 2137–2144.
[10] W. Choi, S. Savarese, Multiple target tracking in world coordinate with single, minimally calibrated camera, European Conference on Computer Vision, 2010, pp. 553–567.
[11] A. Ess, B. Leibe, K. Schindler, L.J. Van Gool, Robust multiperson tracking from a mobile platform, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 31, 10, 2009, pp. 1831–1846.
[12] Q. Zhu, S. Avidan, M.C. Yeh, K.T. Cheng, Fast human detection using a cascade of histograms of oriented gradients, IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, 2006, pp. 1491–1498.
[13] K. Crammer, Y. Singer, On the algorithmic implementation of multi-class SVMs, J. Mach. Learn. Res. 2 (2002) 265–292.
[14] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.D. Collin, B. Kegl, MultiBoost: a multi-purpose boosting package, J. Mach. Learn. Res. 13 (2012) 549–553.
[15] R. Rosipal, N. Kramer, Overview and recent advances in partial least squares, Lect. Notes Comput. Sci 3940 (2006) 34–51.
[16] W.R. Schwartz, A. Kembhavi, D. Harwood, L.S. Davis, Human detection using partial least squares analysis, IEEE Int. Conf. on Computer Vision, 2009, pp. 24–31.
[17] H. Wold, Soft modeling by latent variables: the nonlinear iterative partial least squares approach, Perspectives in Probability and Statistics, 1975, pp. 520–540.
[18] S. Wold, W. Johansson, M. Cocchi, PLS—partial least squares projections to latent structures, 3D QSAR in Drug Design, vol. 1, 1993, pp. 523–550.
[19] A. Bosch, A. Zisserman, X. Muoz, Image classification using Random Forests and Ferns, IEEE Int. Conf. on Computer Vision, vol.1, 8, 2007, pp. 14–21.
[20] S.J. Julier, J.K. Uhlmann, 401-422, Unscented filtering and nonlinear estimation, Proc. The IEEE, vol. 92, 3, 2004, pp. 401–422.
[21] D. Baltieri, R. Vezzani, R. Cucchiara, People orientation recognition by mixtures of wrapped distributions on random trees, Proc. European Conf. on Computer Vision, vol. 7576, 2012, pp. 270–283.
[22] R. Genuer, J.M. Poggi, C.T. Mallot, Variable selection using random forests, Pattern Recognition Letters, vol. 31, 14, 2010, pp. 2225–2236.
[23] K.J. Archer, R.V. Kimes, Empirical characterization of random forest variable importance measures, Computational Statistics and Data Analysis, vol. 52, 4, 2008, pp. 2249–2260.