Computer Vision and Image Understanding 117 (2013) 1502–1511

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

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

Head direction estimation from low resolution images with scene adaptation ☆

Isarun Chamveha [a,*], Yusuke Sugano [a], Daisuke Sugimura [a], Teera Siriteerakul [a], Takahiro Okabe [a], Yoichi Sato [a], Akihiro Sugimoto [b]

[a] *The University of Tokyo, 4-6-1 Komaba, Meguro-Ku, Tokyo 153-8505, Japan*
[b] *National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo 101-8430, Japan*

ARTICLE INFO

Article history:
Received 20 June 2012
Accepted 12 June 2013
Available online 21 June 2013

Keywords:
Head direction estimation
Low resolution image
Appearance-based approach
Scene adaptation
Graph-based image segmentation
Unsupervised learning

ABSTRACT

This paper presents an appearance-based method for estimating head direction that automatically adapts to individual scenes. Appearance-based estimation methods usually require a ground-truth dataset taken from a scene that is similar to test video sequences. However, it is almost impossible to acquire many manually labeled head images for each scene. We introduce an approach that automatically aggregates labeled head images by inferring head direction labels from walking direction. Furthermore, in order to deal with large variations that occur in head appearance even within the same scene, we introduce an approach that segments a scene into multiple regions according to the similarity of head appearances. Experimental results demonstrate that our proposed method achieved higher accuracy in head direction estimation than conventional approaches that use a scene-independent generic dataset.

1. Introduction

Estimating the human visual focus of attention has recently become a popular research trend, as such research has numerous applications in our daily life. For example, it can be used to estimate the attention of people walking along the street [1,2]. Having attention information enables us to easily infer interaction between people and to consequently analyze human interaction or identify on-going activities without requiring any human assistance [3,4]. Head direction is known to be an important factor in inferring the focus of attention of humans. Therefore, techniques for estimating head direction are considered important and have attracted great interest recently.

Various image-based approaches have been proposed for estimating eye gaze. However, most of them[1] require high resolution images [6,7] or special equipment such as depth cameras [8,9] or actively controlled pan-tilt-zoom cameras [10]. Hence, one of the major remaining technical challenges is how to estimate human gaze or head direction with low resolution images. In some application scenarios such as visual surveillance, the head regions in input images are often quite small. Small images contain limited information; accurately estimating head direction in such cases remains a challenging task.

The use of appearance-based approaches is thought to be promising for estimating head directions from low resolution images. Compared with model-based methods such as active appearance models [11,12], which rely on geometric facial models and require localization of facial elements, appearance-based methods directly use pixel values of an image as an input to extract image features and are known to be effective even with low resolution images.

Appearance-based head direction estimation approaches rely heavily on a dataset used for training estimators. This is due to the fact that head appearances can change significantly from scene to scene. Even in the same scene, there could be substantial differences in head appearance due to extreme differences in illumination or camera viewing angle. Therefore, a training dataset is best taken from the same location as the target data. However, collecting ground-truth training samples is a labor-intensive and time-consuming task, and it is prohibitively expensive to collect ground-truth data manually every time a head direction estimation method is applied to different scenes.

We propose an appearance-based head direction estimation method in order to overcome these problems. Our method is based on two key ideas: automatically acquiring a training image dataset with ground-truth head directions and segmenting a scene into multiple regions with similar head appearances. We first construct a training image dataset by tracking pedestrians in a scene of interest to capture head images where their walking directions are re-

☆ This paper has been recommended for acceptance by Jordi Gonzàlez.
* Corresponding author.
E-mail address: isarunc@iis.u-tokyo.ac.jp (I. Chamveha).
[1] We refer readers to [5] for a recent survey on approaches to head pose estimation.

1077-3142/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2013.06.005

garded as a ground truth head orientation. To address the problem of appearance differences within a scene, the scene is segmented into multiple regions based on the similarity of head appearances, and a head direction estimator is then trained for each region. This approach enables us to test each head image with the estimator trained with data taken from the same region. Higher accuracy can thus be expected because the data used to train the estimator have a similar appearance to the test data.

This paper is organized as follows. Section 2 describes related works on head direction estimation from low resolution images. Section 3 explains the details of our proposed framework. Section 4 describes the method to automatically acquire a scene-specific dataset. Section 5 introduces an adaptive scene segmentation method that solves the problem of appearance differences within a scene. We describe detailed experiments and a thorough analysis of the proposed method in Section 6, and we conclude the paper in Section 7.

2. Related work

Many attempts have been made recently to estimate head directions from low resolution images. Robertson and Reid [13] used skin color as a descriptor and a binary tree algorithm to construct the head direction classifier. Body direction is also used to filter out poses that are not physically plausible. Benfold and Reid [14] proposed a descriptor that learns a model of skin color automatically and used randomized ferns for head direction estimation. Orozco et al. [15] proposed an image descriptor using similarity distance maps with class-mean appearance templates, and a multi-class Support Vector Machine (SVM) for head direction classification. Benfold and Reid [1] utilized pedestrian tracking to accurately locate head position and perform head pose estimation. Their approach tracks pedestrians using a Kalman filter based tracking method. The head directions of the pedestrians are then estimated using a randomized ferns classifier with the histogram of oriented gradients (HOGs) features and color triplets comparisons (CTCs) as fern decisions. Schulz et al. [16] proposed an approach that integrates pedestrian head localization and head pose estimation to achieve high head pose estimation accuracy. Schulz and Stiefelhagen [17] trained eight head pose detectors, one for each pose class, to detect pedestrian heads. Their approach also integrates head pose predictions over time using particle filtering to achieve improved robustness and efficiency.

These studies used head images with resolution as low as 20×20 pixels for training and testing the classifiers. While they are shown to work well for low resolution head images, they suffer from one important problem: a large number of training images with ground-truth labels, i.e., correct head orientations, are required. Orozco et al. [15] used 800 manually cropped head images, 100 for each direction class from the i-LIDS [18] dataset. Gourier et al. [19] turned downsampled images from the Pointing'04 dataset into low resolution 23×30 dimension images. Robertson et al. [20,13] used ground-truth samples produced by a human user drawing the line-of-sight of pedestrians in the images. Schulz et al. [16] used 7675 positive head pose samples and a set of negative non-head samples to construct the head pose classifier.

The work that is most relevant to our proposed method is that of Benfold and Reid [21]. They proposed an approach to solve the problem of a lack of training data. They constructed an unsupervised head direction estimator using a conditional random field model based on the same premise that people turn their head to where they are walking. However, their approach requires tracking information of the test data, which are sometimes unavailable such as in still images. Another unsupervised approach was recently proposed by Chen and Odobez [22]. Their approach jointly estimates the body pose together with the head pose, and this makes the method more robust by filtering out physically impossible head poses. However, both of these two methods do not consider appearance differences within the same scene.

3. Proposed framework

Appearance-based head direction estimation involves determining a head direction p from a feature vector $\boldsymbol{h}$ of an input head image. We define p as the head direction in an image plane. In our work, head direction estimation is defined as a regression task, where head directions are defined as continuous angles as illustrated in Fig. 1.

With a set of training samples $D = \{(\boldsymbol{h}_k, p_k)\}$, the mapping $p = f(\boldsymbol{h})$ between the head direction and the feature vector can be learned through various regression algorithms. The mapping function then can be used to estimate a head direction p^* from a new feature vector $\boldsymbol{h}^*$ in test scenes.

As discussed above, an important problem yet largely ignored in previous studies is how to obtain appropriate training samples D. Since we implicitly assume the mapping function $f(\boldsymbol{h})$ is identical in both training and test datasets, estimation accuracy highly depends on how similar the head images are in both datasets. Due to various factors such as lighting conditions or camera positions, head appearances in different scenes and even within the same scene can be significantly different. An example of such differences in appearance of people within the same scene is shown in Fig. 2. Even though pedestrians are walking in the same direction, their head appearances are different when captured from different locations. In other words, if lighting conditions or camera positions are significantly different between the locations where training and test images are taken, mappings between the direction and the appearance would also be different. Nevertheless, it is not always possible to collect training samples for every test case.

The framework of our proposed method is summarized in Fig. 3. Our method acquires training data from an input video sequence by using walking directions as a cue to infer head directions. The scene is then segmented into multiple regions with similar head appearances, and a head direction estimator is constructed for each region.

In order to obtain walking trajectories of pedestrians in the video, we employed the head tracking method by Benfold and Reid [1]. The method is based on a Kalman filter [23] with two types of measurements: the head locations given by a HOG-based head detector [24] and the velocity of head motion computed from mul-

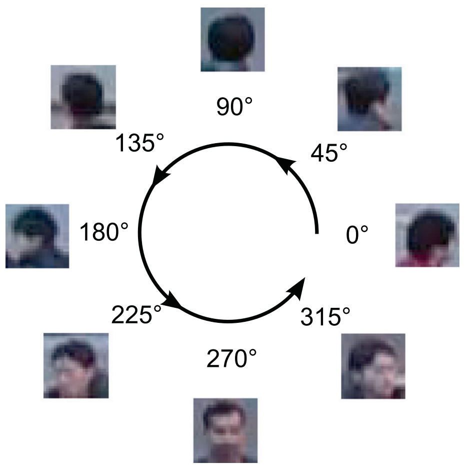

Fig. 1. Illustration of a head direction regression task in which head directions are defined as continuous angle values in an image plane.

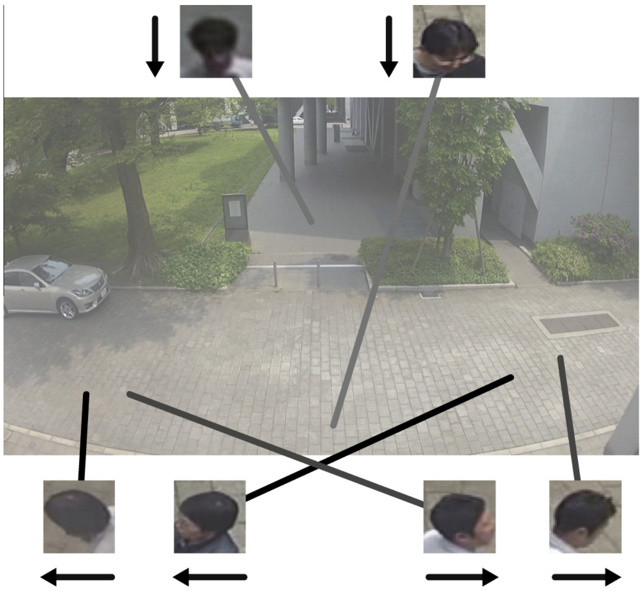

Fig. 2. An example of differences in appearance of people in a scene. Even when the pedestrians are walking in the same direction, the head appearance is different when captured from different locations in the scene.

tiple corner features [25,26]. In each frame, a head image I, a head location $\boldsymbol{u} = (x, y)$ in the image plane, and a measurement error $\boldsymbol{c} = (c^{(x)}, c^{(y)})$ are collected for analysis, where the terms $c^{(x)}$ and $c^{(y)}$ are the respective variances of the measurement on x and y axes of the Kalman filter. The pedestrian tracking algorithm is applied to the entire input sequence, and a trajectory, i.e., a set of head images $\{I_1, \ldots, I_N\}$, head locations $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}$ and error measurements $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$, is acquired for each pedestrian. Here, N denotes the length of the trajectory and it varies depending on the trajectory.

4. Training data acquisition

This section describes our technique to aggregate a scene-specific dataset. Given tracked trajectories of pedestrians, we estimate their walking direction, which can be assumed to indicate their head directions in the images. Erroneous samples that will cause errors in the trained estimators are rejected, and then we collect the remaining training samples to construct the head direction dataset. The proposed method is described in more detail in the following sections.

4.1. Estimating walking directions

To account for the fact that pedestrians do not always walk straight, our method first divides each possibly curved trajectory into straight line segments. More specifically, each trajectory S is divided into segments $\{S_1, \ldots, S_M\}$ by polyline simplification using the Douglas–Peucker algorithm [27]. This algorithm constructs a minimal set of lines so that the orthogonal distances from each point to its nearest line is less than a given threshold $d_{\max}$. Since pedestrians get to appear smaller as they move away from the camera, the threshold $d_{\max}$ should be defined in a location-dependent way. Therefore, based on the fact that the physical size of the curve is proportional to the observed head size, we define the threshold $d_{\max}$ as $d_{\max} = \tau_p \cdot \bar{s}_t$, where τ_p is a scale-invariant constant and $\bar{s}_t = \sum_{i=1}^{N} \sqrt{s_x(\boldsymbol{u}_i) \cdot s_y(\boldsymbol{u}_i)}/N$ is the average head length calculated assuming a square shape observed over a trajectory. $s_x(\boldsymbol{u})$ and $s_y(\boldsymbol{u})$ are the expected width and height of the head at the position $\boldsymbol{u}$. They are calculated assuming that the average human height is 1.7 m, and heads are modeled as cylinders that are 22.0 cm tall and 20.0 cm in diameter, in the same manner as [1]. An example of polyline simplification is shown in Fig. 4. In the figure, the curved line shows the raw tracking result, and the straight lines show line segments obtained using the polyline simplification algorithm.

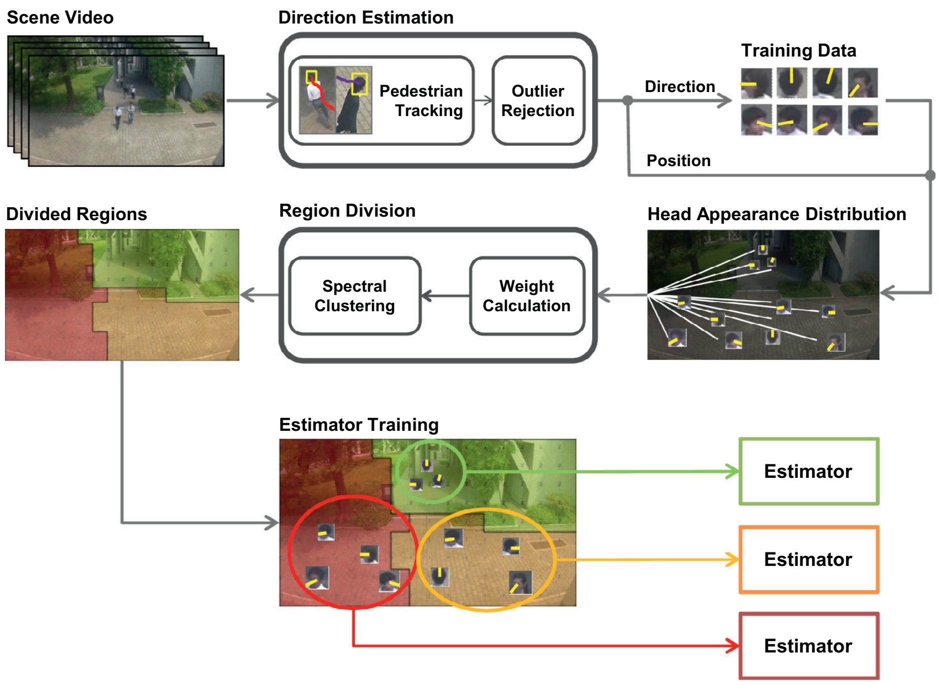

Fig. 3. The proposed framework. From an input video sequence, our method acquires head direction training data by using walking directions as a cue to infer head directions. The scene is then segmented into multiple regions with similar head appearance on the basis of acquired training samples, and head direction estimators are constructed separately for each region.

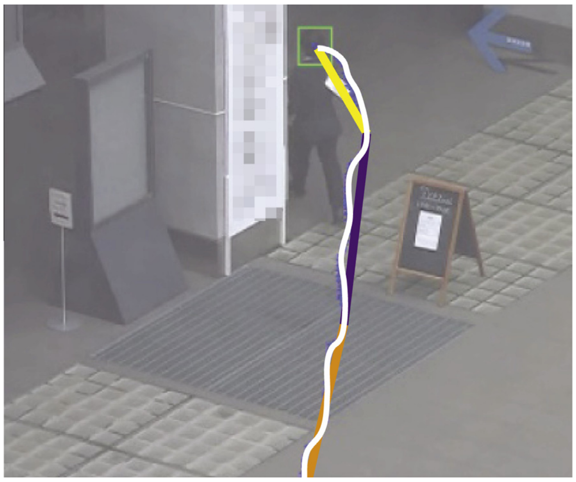
Fig. 4. An example of polyline simplification. The curved line shows a tracking result and straight lines show the result of polyline simplification.

4.2. Rejecting outlier segments

Walking directions obtained from polyline simplification of a trajectory do not always correspond to head orientations since people can move their heads freely even while they are walking. This brings errors in the training labels. Head direction estimation algorithms are not always robust to such outliers, and thus it is preferable to reject them prior to the learning stage. To address this problem, we introduce a strategy to reject unreliable segments from the tracking results.

There are three kinds of segments that cause erroneous training samples: (1) segments with large tracking errors, (2) segments with short length or slow movement, and (3) segments with large image variance. The details of each kind are as follows. Let us assume that a segment contains T head locations $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\}$.

4.2.1. Segments with large tracking errors

Although the pedestrian trackers can resume their tracking and are robust to a few mis-detections, a large number of mis-detections can produce erroneous trajectories and poor head image localizations. These situations will result in a large number of erroneous points and large line fitting errors, which should be rejected.

To calculate the number of erroneous points, a point $\boldsymbol{u}_t$ is identified to be erroneous if the error measurement of the tracker is significantly large:

$$\sqrt{\left(\frac{c_t^{(x)}}{s_x(\boldsymbol{u}_t)}\right)^2 + \left(\frac{c_t^{(y)}}{s_y(\boldsymbol{u}_t)}\right)^2} > \alpha. \tag{1}$$

where $\boldsymbol{c}_t = \left(c_t^{(x)}, c_t^{(y)}\right)$ is the measurement error of the tracker. Since these measurement errors should be evaluated according to their physical size, head width $s_x(\boldsymbol{u}_t)$ and height $s_y(\boldsymbol{u}_t)$ at the location of the tracker $\boldsymbol{u}_t$ are introduced to scale the measurements.

Using this measure, we reject segments if the ratio of erroneous segment points to the total number of points in the segment is larger than a predefined threshold τ_e. Note that α indicates the acceptable error level, while τ_e controls the number of acceptable erroneous points in the segment. These parameters are not independent to each other, thus we first chose α to reject trajectory points where the head detector failed. When the head detector of the tracker module failed twice in a row, α is set to the error level of the tracker at that moment. After α is selected, τ_e is then chosen accordingly.

To reject segments with large line fitting errors, we calculate the summation of the orthogonal distances from each point to the estimated line over a segment and divide the summation by the length of the segment. We then reject a segment if

$$\frac{1}{|\boldsymbol{u}_T - \boldsymbol{u}_1|} \cdot \sum_{t=1}^{T} \frac{|a \cdot x_t + b \cdot y_t + c|}{\sqrt{a^2 + b^2}} \geqslant \tau_l, \tag{2}$$

where τ_l is a threshold indicating the maximum acceptable level of line fitting errors. $\boldsymbol{u}_t = (x_t, y_t)$ is a point in the segment. The left-hand side of Eq. (2) is a scale-independent line fitting error of the estimated line $ax + by + c = 0$.

4.2.2. Segments with short length or slow movement

Pedestrians making slow or no movements are often seen in a scene, e.g., people talking to each other in the same location. Using the walking direction to estimate head directions in such situations would give erroneous results. Therefore, segments that are short in length or that have slow movements need to be rejected. Rejecting segments with short length also removes cases where false positive objects are detected as heads, which usually stay within a small area. Therefore, we reject a segment if

$$\frac{|\boldsymbol{u}_T - \boldsymbol{u}_1|}{\bar{s}_s} \leqslant \tau_n \quad \text{or} \quad \frac{|\boldsymbol{u}_T - \boldsymbol{u}_1|}{T \cdot \bar{s}_s} \leqslant \tau_v \tag{3}$$

where τ_n and τ_v are predefined thresholds for detection of segments with short length and slow movements, respectively, and $\bar{s}_s = \sum_{i=1}^{T} \sqrt{s_x(\boldsymbol{u}_i) \cdot s_y(\boldsymbol{u}_i)}/T$ is the average of the head-length factors over the segment.

4.2.3. Segments with large image variance

Pedestrians in the video are sometimes observed turning their head while they walk, which also leads to erroneous direction estimation results. Because large variations in head appearance are expected in such cases, segments with large image variations should be rejected. We calculate the variance of resized head image vectors $\{\widehat{\mathbf{I}}\}$ whose dimensions are denoted by C. A segment is considered to have large variance if

$$\frac{\sum_{t=1}^{T} |\widehat{\mathbf{I}}_t - \bar{\mathbf{I}}|^2}{T \cdot C} \geqslant \tau_{var}, \tag{4}$$

where $\widehat{\mathbf{I}}_t$ denotes the tth resized image, $\bar{\mathbf{I}}$ is a mean image calculated from all resized images $\widehat{\mathbf{I}}$ in the segment, and τ_{var} is a predefined constant. While high τ_{var} makes the algorithm accept more samples, low τ_{var} makes the algorithm more selective about the stability of head images.

4.3. Representative image selection

Most outlier segments are rejected in the outlier segment rejection process, and the remaining segments contain correct data. Since only one orientation is assigned to each segment, most of the images in each segment are redundant. Using all the images for training would result in an excessively large dataset, which increases the computational time for many machine learning tools. Therefore, one representative image per segment is selected and used as training data.

In this work, we select the image that is most similar to the mean image of the segment. For each segment, the Mahalanobis distance from the mean image is calculated for every resized image $\widehat{\mathbf{I}}_t$ in the segment and the image with the lowest distance is selected. This enables us to select the representative image while avoiding effects that can be seen in the mean image, e.g., blur or distortion.

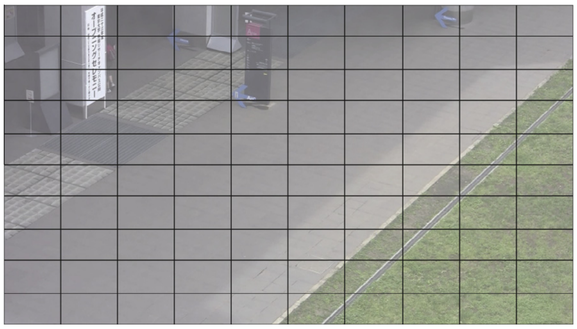

Fig. 5. An example of dividing a scene into 10×10 unit regions ($K = 100$). These regions serves as the smallest unit to construct each region.

5. Adaptive scene segmentation for localized direction estimation

As mentioned before, appearance differences of training samples in the scene can reduce the accuracy of the estimator. This section addresses our approach of segmenting a scene into multiple regions in each of which the heads with the same direction have a similar appearance. Because there is no definitive way to define regions with similar head appearances, an unsupervised clustering approach is taken to segment a scene into such regions. In this work, spectral clustering is used to segment a scene. Given a set of points and a similarity matrix defining the similarity of each pair of points, spectral clustering techniques cluster the set into disjoint subsets with high intra-cluster similarity and low inter-cluster similarity. Normalized cut [28], which is one of the most common spectral clustering algorithms, is applied in our approach.

Our approach first divides the scene into K rectangular unit regions $V = \{v_1, \ldots, v_K\}$ which are used as the set of nodes. Fig. 5 shows an example of 10×10 unit regions ($K = 100$). Then normalized cut is applied to cluster the regions V into R clusters, $\mathbf{A} = \{A_1, \ldots, A_R\}$, where $A_i \neq \emptyset$, $A_i \subset V$, $A_i \cap A_j = \emptyset$ $(1 \leqslant \forall i,j \leqslant R, i \neq j)$ and $\bigcup_{i=1}^{R} A_i = V$. In the following sections, we discuss how to calculate the similarity weight function $w(v_i, v_j)$ for each pair of unit regions and how to choose the appropriate number of regions.

5.1. Weight function

With the dataset D obtained using the method described in Section 4, we define D_v as training samples captured at the unit region v. Our proposed similarity weight $w(v_i, v_j)$ between two unit regions v_i and v_j is defined with the distance weight w_d and the sample weight w_s as $w(v_i, v_j) = w_d(v_i, v_j) \cdot w_s(v_i, v_j)$.

The distance weight, $w_d(v_i, v_j)$, measures how close two unit regions v_i and v_j are. This takes into account the fact that training samples acquired from nearby locations tend to be more similar than those acquired from distant locations. The distance weight also makes segmented regions spatially smooth. We define distance weight w_d as follows:

$$w_d(v_i, v_j) = e^{-\frac{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2}{\sigma_d}}, \tag{5}$$

where $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are the positions of the unit regions v_i and v_j, respectively and σ_d is a predefined constant.

Sample weight $w_s(v_i, v_j)$ measures the similarity between training samples acquired from the two unit regions v_i and v_j. Two unit regions v_i and v_j should be merged into the same region if the training samples D_{v_i} are similar to D_{v_j}. Sample weight is defined as

$$w_s(v_i, v_j) = e^{-\frac{d_u(v_i, v_j)}{\sigma_s}}, \tag{6}$$

where σ_s is a constant and $d_u(v_i, v_j)$ is a function that measures the difference between samples in two unit regions. The comparison is done between training samples corresponding to similar head directions, i.e.,

$$d_u(v_i, v_j) = \frac{\sum_{(\boldsymbol{h}_i, p_i) \in D_{v_i}, (\boldsymbol{h}_j, p_j) \in D_{v_j}} d(\boldsymbol{h}_i, \boldsymbol{h}_j) \cdot \phi(p_i, p_j)}{\sum_{(\boldsymbol{h}_i, p_i) \in D_{v_i}, (\boldsymbol{h}_j, p_j) \in D_{v_j}} \phi(p_i, p_j)}, \tag{7}$$

where $(\boldsymbol{h}_i, p_i)$ and $(\boldsymbol{h}_j, p_j)$ are the feature vectors and head direction labels for a training sample in the dataset D_{v_i} and D_{v_j}, respectively. Here, $\phi(p_i, p_j)$ is defined using a threshold[2] θ:

$$\phi(p_i, p_j) = \begin{cases} 1 & \text{if } |p_i - p_j| < \theta \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The value $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$ measures differences between a pair of samples, and is defined as the weighted distance between the feature vectors:

$$d(\boldsymbol{h}_i, \boldsymbol{h}_j) = \sqrt{(\boldsymbol{h}_i - \boldsymbol{h}_j)^T M (\boldsymbol{h}_i - \boldsymbol{h}_j)}, \tag{9}$$

where $M = \text{diag}[M_i]$ is the diagonal matrix indicating the importance of each feature in the feature vectors. M_i should be large if the ith feature has a strong impact on distinguishing head directions. Although the importance matrix M can be obtained by using several approaches, in this work, M was obtained from the variable importance vector calculated from the random trees estimator [29], which was trained using the whole dataset D.

5.2. Determining the number of regions

In addition to the weight function w, it is also important to select the appropriate number of regions. It is preferable for a scene to be segmented into as many regions as possible to take advantage of having samples with similar appearances inside the same region. However, if a region is too small, the number of training samples will be insufficient, and the trained estimators will have significant generalization errors.

We perform cross validation on the scene segmented with different numbers of regions and select the one that minimizes the cross-validation error. The cross validation errors is defined as the weighted sum of the validations errors in each region: for a segmentation $\boldsymbol{A} = \{A_1, A_2, \ldots, A_R\}$,

$$E_g(R) = \frac{1}{|D|} \sum_{r=1}^{R} E_c(D_r) \cdot |D_r|, \tag{10}$$

where D_r is the set of training samples captured within region A_r, and $E_c(D_r)$ is the 5-fold cross validation error using the training data D_r. For each sequence, cross-validation errors for scene segmentation with $R (1 \leqslant R \leqslant R_{\max})$ are calculated, and the number R^* that minimizes the cross-validation error is then selected, i.e., $R^* = \arg\min_R E_g(R)$. We consider head directions estimated by using our proposed method in Section 4 as ground-truth data, and therefore we do not use manually-labeled ground truth data for computing cross-validation errors.

5.3. Training estimators

As a result of the above processes, we obtain a set of regions $\boldsymbol{A}' = \{A'_1, \ldots, A'_{R^*}\}$ in each of which the appearance of the training samples is similar. Estimators $f_1, \ldots, f_{R^*}$ are then created for each region, and each of them is trained with the samples in its corre-

[2] In our experiments, head samples with differences less than 45.0° were defined as being similar, i.e., we set $\theta = 45.0$ (°).

sponding region; i.e., estimator f_j for region A'_j is trained with the dataset $D_{train,j} = \bigcup \left\{ D_v; v \in A'_j \right\}$. The estimator in each region is applied for test samples captured in its corresponding region; i.e., the test samples in region A'_j are tested with the estimator f_j. Note that test samples are separated from training samples and are not included in the dataset D.

6. Experimental results

We conducted experiments using five video sequences that were recorded using different cameras in different scenes. The details of each sequence and the numbers of samples obtained as a result of the training data acquisition approach are summarized in Table 1. Example frames in the videos are shown in Fig. 6. Examples of the obtained head images are also shown with the estimated walking direction shown next to the image.

The parameters were set as follows for every scene; $\tau_p = 0.8$, $\alpha = 0.5$, $\tau_l = 5$, $\tau_e = 0.4$, $\sigma_d = 1000$, $\sigma_s = 0.1$, $\tau_n = 3.0$, $\tau_v = 0.02$, $\tau_{var} = 0.0035$. The effect of applying the rejection methods is analyzed in Section 6.3, and the robustness against the parameter setting is analyzed in Section 6.4.

6.1. Estimation of head directions

To evaluate the performance of our proposed method, we compared our method with regressors trained with a generic dataset that consists of head images collected from other scenes. We constructed a generic dataset using 1477 training samples taken from the Gaze Direction Dataset [30], which was used in [1]. Fig. 7 shows examples of head images included in the generic dataset.

All of the head images were resized to 40×40 pixels ($C = 1600$), and all of the scenes were divided into 16×9 unit regions ($K = 144$). An image descriptor similar to the one in [21] was used here. The descriptor is the concatenation of two features. The first feature measures color difference between two pixels at two different locations. The second feature measures difference between two different bins from Histograms of Gradients (HOGs) features extracted from the head images divided into 4×4 cell grids and normalized spatially across 2×2 blocks of cells. In our work, 400 pairs of points were chosen randomly for each feature.

The experiment was conducted using two regressors: Support Vector Regression (SVR) and regression with random trees [29]. Both of them were implemented by using OpenCV library [31]. SVR is one of the most common machine learning tools used in head direction and gaze estimation [15,32]. The combination of random trees and the above-discussed descriptor is similar to the estimator in [21] and was used for a fair comparison between our results with those reported in [21]. Our method finished training within 10 min, and testing took less than 1 ms per test sample on an Intel Core 2 Duo 3.00 GHz CPU.

Table 1
Details of sequences used in our experiments. The first three columns show the name, resolution and length of each sequence, respectively. The fourth column indicates the number of test samples captured from each sequence, and the last column shows the number of training samples acquired with the proposed method.

Sequence name	Resolution	Length (min)	Test samples	Obtained samples
Sequence 1	1920×1080	420	300	7841
Sequence 2	1120×780	10	100	1312
Sequence 3	1280×720	10	135	693
Sequence 4	1920×1080	90	305	3075
Town Centre[a]	1920×1080	22	4347	6190

[a] The Town Centre sequence was the publicly available sequence from [21].

A comparison of the mean absolute angle errors (MAAEs) between our method and the baseline using the generic dataset is summarized in Fig. 8. Here, we also compared the results without using the scene segmentation method. The *Generic* results were calculated based on regressors trained using the generic dataset, the *Undivided* results were calculated based on regressors trained using samples acquired without scene segmentation, and *Proposed* results were calculated based on our proposed method. *Benfold* result shows the angle error stated in [21].

The graphs show that the errors in regression tasks using the dataset obtained with our method are significantly smaller than those of the generic dataset. Scene segmentation further reduces errors for scenes with large variations in lighting conditions, such as sequence 3, or large differences in camera viewing angles such as sequence 4. Our result is comparable to that of Benfold and Reid [21].

6.2. Adaptive scene segmentation

To test the effectiveness of region segmentation in our method, we measured the relation between cross-validation errors and actual estimation errors. We applied our method with different numbers of regions and recorded their respective cross validation errors. In our experiments, we set the maximum number of regions to calculate cross-validation errors to 10. The comparison of estimation errors and cross-validation errors for SVR with different number of regions are shown in Fig. 9. Both the cross-validation errors and the estimation errors increase when the number of regions increases to more than 5 and have been omitted from the graph for clearer representation. The number of regions that minimizes cross validation errors was chosen as the optimum number of regions. It can be seen that minimizing the cross-validation errors on training samples also minimize the estimation errors on test samples. The results of scene segmentation are shown in Fig. 10. It can be seen that in sequences 3 and 4, areas with a large camera angle or illumination differences were segmented automatically. No significant change in performance was seen in the other sequences where head appearances remain relatively uniform in the scene.

6.3. Analysis of outlier rejection

To measure the effectiveness of our outlier rejection rules described in Section 4.2, we tested our proposed method with omitting each rule. An example result from sequence 4 is shown in Fig. 11. Similar trends were observed in the other datasets, although we did not include those results here. It can be seen that our proposed method achieves the best estimation accuracy while maintaining the smallest dataset size.

These results indicate that short segment length and slow movement criteria significantly reduce estimation errors. This is intuitively reasonable because these rules reject trajectories where pedestrians are talking to each other, which are often observed in scenes. Rejection of short length segments also further reduces the errors by rejecting trajectories generated by false positive objects. In addition, it can be seen that rejection of samples with high variance significantly reduces the number of captured samples. This improves the training speed for large datasets.

6.4. Analysis of parameter settings

In this section, the effects of different parameter values on the results are analyzed. We conducted experiments by applying the proposed method and changing each parameter value by 20%. We did not perform the analysis on the Town Centre dataset be-

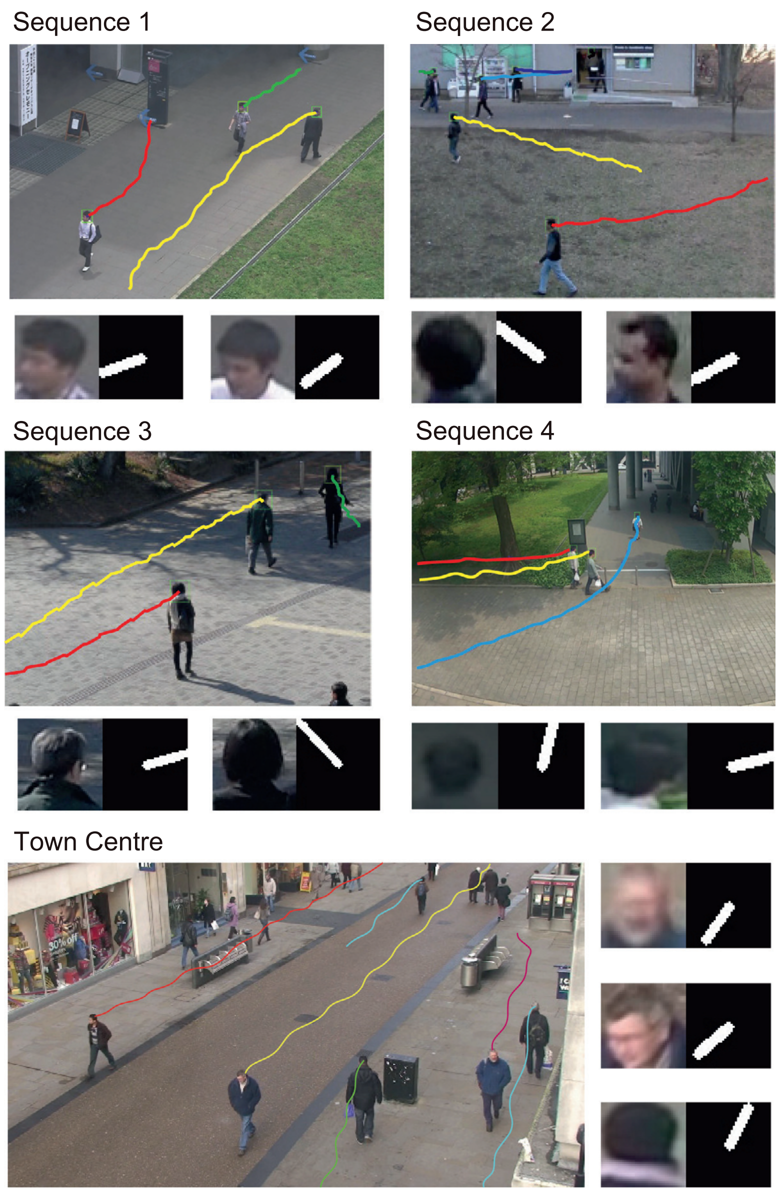

Fig. 6. Example frames in test video sequences. The input video frames are overlaid with pedestrian tracking results. Examples of obtained head images are also shown; the white line in the right part of each image represents the estimated walking direction.

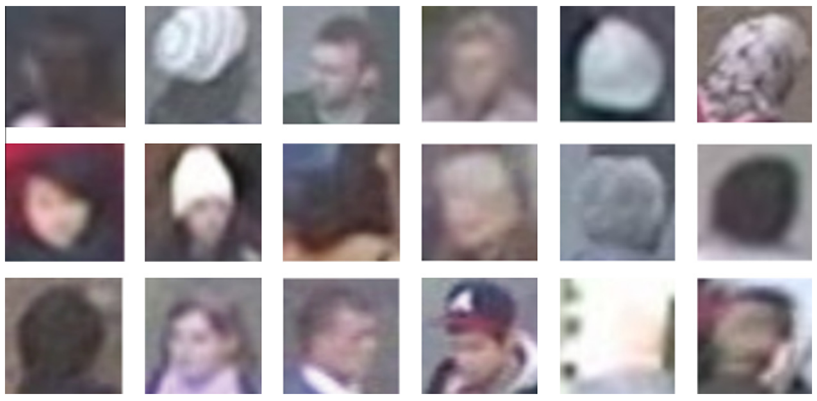

Fig. 7. Example images in the Gaze Direction Dataset.

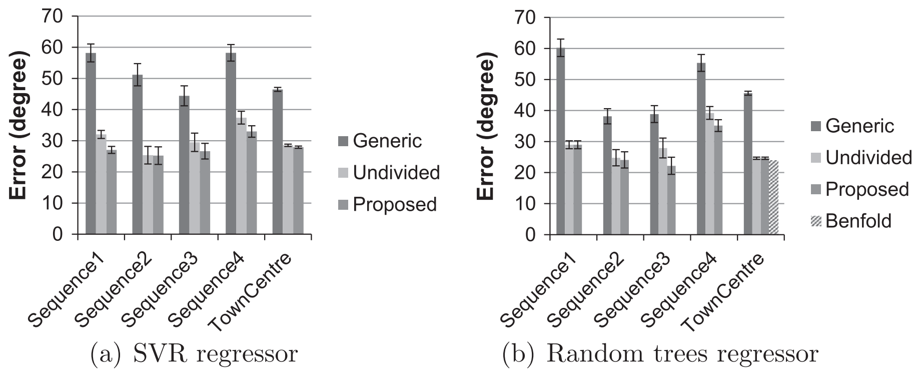

Fig. 8. Errors of head direction estimation for SVR and random trees regressors. *Generic* results were calculated based on regressors trained using the generic dataset; *Undivided* results were calculated based on our sample collection approach without scene segmentation; *Proposed* results were calculated based on regressors trained using samples acquired with our method, and *Benfold* result shows the angle error stated by Benfold and Reid [21] on Town Centre dataset using their proposed method. The errors were measured using the mean absolute angle error (MAAE). Standard errors are indicated as error bars.

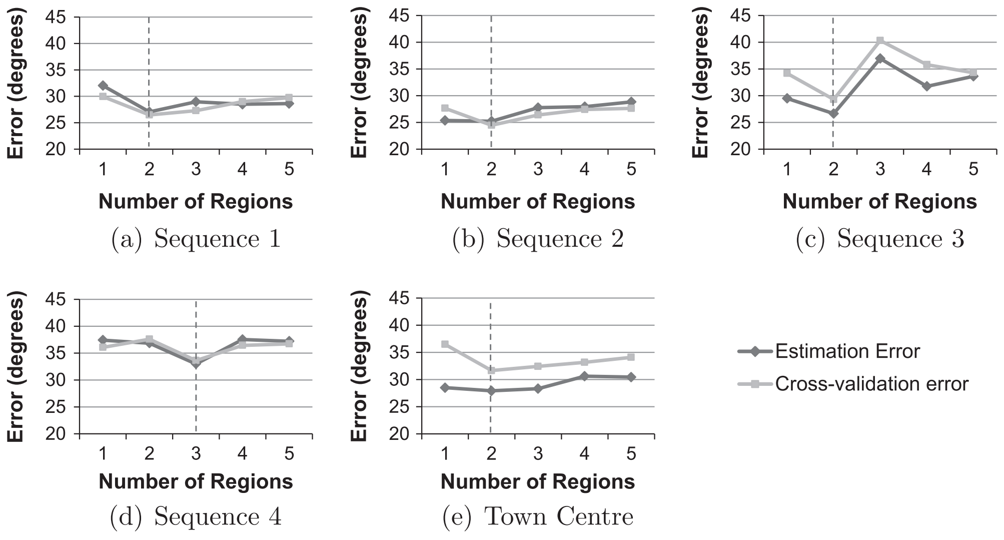

Fig. 9. Comparison of estimation errors and cross-validation errors with varying number of region. The values of R^* where cross validation errors were minimum were selected and are shown as dotted lines in the graphs.

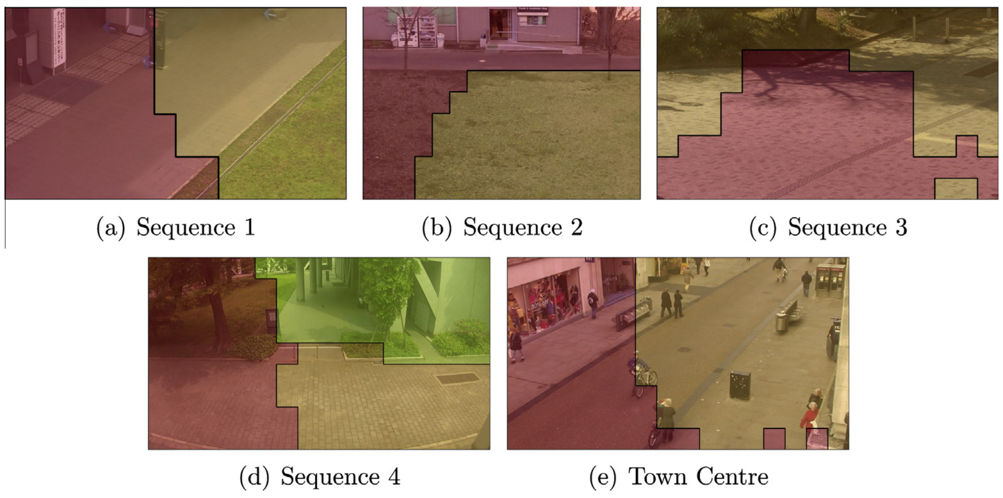

Fig. 10. Scene segmentation results with the proposed method. The figure is best viewed in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

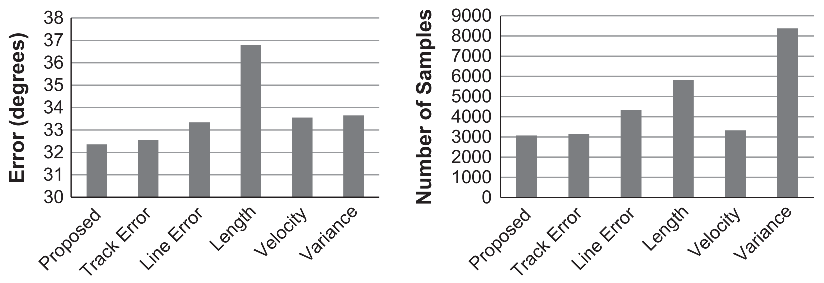

Fig. 11. Errors and number of samples captured from sequence 4 with various outlier rejection methods omitted using SVR as regressor. *Proposed* column shows our proposed method. *Track Error* and *Line Error* columns omitted the rejection of segments with erroneous tracking and large line fitting errors, respectively. *Length* and *Velocity* columns omitted the rejection of segments with short length and slow movement, respectively. *Variance* column omitted the rejection of segments with high image variance.

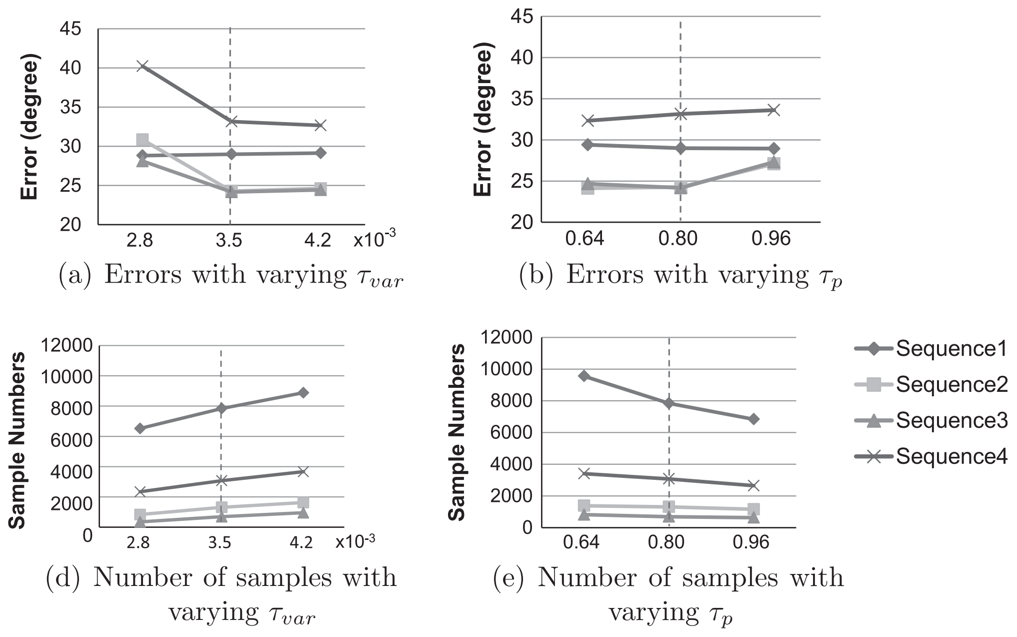

Fig. 12. Variations of estimation errors and number of collected samples with varying parameters. Center columns with dotted lines indicate the default value of each parameter.

cause the tracking results provided by the authors were used, and tracker error variance values were not available.

We show two example results of parameter tests in Fig. 12. In the figure, the center columns with dotted lines show the default value mentioned in Section 6. The left and right columns show the default values that were changed by 20%. In each experiment, only one parameter was changed, and the other parameters were kept at their default values. Generally speaking, if the parameters are set too strictly, estimation errors increase due to the lack of sample variations. If the parameters are set to be more tolerant, the number of samples increases while the estimation errors are not significantly reduced.

Although rejecting segments with large variance reduces estimation errors as stated in Section 6.3, it is apparent that estimation errors significantly increase with a stricter threshold. This indicates that although image variance helps in rejecting images with incorrect head directions, variations in head appearance are also important for training regressors. Increasing the threshold value by 20%, however, did not significantly affect the estimation result. This is because image segments where people turn their head usually have large variance, and thus, there is a large margin for the variance threshold to reject such segments. Increasing the polyline fitting threshold will cause curved lines to be estimated as straight lines. This significantly increased estimation errors, especially in sequences 2 and 3 which contain a small number of samples. Reducing the threshold increased the number of samples but did not significantly reduce estimation errors. This is because if the line estimated by polyline simplification is sufficiently straight, reducing the threshold will further divide the line but will not yield any benefits.

7. Conclusion

We proposed a method of appearance-based head direction estimation that can automatically adapt to test scenes. The key idea behind the proposed framework is to use walking directions as a cue to infer head directions. A pedestrian tracker is first applied to the input video sequence, and then head direction for each pedestrian is estimated based on his/her walking direction. Outlier segments are rejected, and then a scene-specific dataset of head images labeled by their walking directions is automatically acquired. Each scene is then segmented into multiple regions according to the appearance of acquired head images with the same direction. Finally, a head direction estimator for each region is created by using training samples acquired from that region. The results of our experiments verified that our method estimates head direction accurately without any need to manually collect a ground-truth dataset in real scenes. This is a significant advantage compared to existing methods when applied to practical scenarios.

Appearance-based head direction estimation from low-resolution images is itself a difficult task, and there is still room for improvement in both feature description and estimation techniques. We believe that investigating the learning algorithm itself is an important future task.

Acknowledgments

This research was supported by JST, CREST.

References

[1] B. Benfold, I. Reid, Guiding visual surveillance by tracking human attention, in: Proc. 20th British Machine Vision Conference (BMVC2009), pp. 14.1–14.11.

[2] K. Smith, S.O. Ba, J.-M. Odobez, D. Gatica-Perez, Tracking the visual focus of attention for a varying number of wandering people, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2008) 1212–1229.

[3] C.-W. Chen, R. Ugarte, C. Wu, H. Aghajan, Discovering social interactions in real work environments, in: Proc. IEEE International Workshop on Social Behavior Analysis 2011 (SBA2011), pp. 933 –938.

[4] S.O. Ba, J.-M. Odobez, Multiperson visual focus of attention from head pose and meeting contextual cues, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 101–116.

[5] E. Murphy-Chutorian, M.M. Trivedi, Head pose estimation in computer vision: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 607–626.

[6] A. Puri, H. Kannan, P. Kalra, Coarse head pose estimation using image abstraction, in: Proc. 9th Conference on Computer and Robot Vision (CRV2012).

[7] D. Huang, M. Storer, F. De la Torre, H. Bischof, Supervised local subspace learning for continuous head pose estimation, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR2011), pp. 2921–2928.

[8] G. Fanelli, T. Weise, J. Gall, L.V. Gool, Real time head pose estimation from consumer depth cameras, in: Proc. 33rd Annual Symposium of the German Association for Pattern Recognition (DAGM2011), pp. 101–110.

[9] G. Fanelli, J. Gall, L. Van Gool, Real time head pose estimation with random regression forests, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR2011), pp. 617–624.

[10] N. Krahnstoever, M.-C. Chang, W. Ge, Gaze and body pose estimation from a distance, in: Proc. 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS2011), pp. 11 –16.

[11] T.F. Cootes, G.J. Edwards, C.J. Taylor, Active appearance models, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 681–685.

[12] I. Matthews, S. Baker, Active appearance models revisited, Int. J. Comput. Vis. 60 (2004) 135–164.

[13] N.M. Robertson, I.D. Reid, Estimating gaze direction from low-resolution faces in video, in: Proc. 9th European Conference on Computer Vision (ECCV2006), pp. 402–415.

[14] B. Benfold, I. Reid, Colour invariant head pose classification in low resolution video, in: Proc. 19th British Machine Vision Conference (BMVC2008).

[15] J. Orozco, S.G. Gong, T. Xiang, Head pose classification in crowded scenes, in: Proc. the 20th British Machine Vision Conference (BMVC2009).

[16] A. Schulz, N. Damer, M. Fischer, R. Stiefelhagen, Combined head localization and head pose estimation for video-based advanced driver assistance systems, in: Proc. 33rd Annual Symposium of the German Association for Pattern Recognition (DAGM2011), pp. 51–60.

[17] A. Schulz, R. Stiefelhagen, Video-based pedestrian head pose estimation for risk assessment, in: 15th International IEEE Conference on Intelligent Transportation Systems (ITSC2012), pp. 1771–1776.

[18] Home Office Scientific Development Branch, Imagery library for intelligent detection systems (i-LIDS), 2007. <http://homeoffice.gov.uk>.

[19] N. Gourier, J. Maisonnasse, D. Hall, J.L. Crowley, Head pose estimation on low resolution images, in: Proc. First International Evaluation Workshop on Classification of Events, Activities and Relationships (CLEAR 2006), pp. 270–280.

[20] N.M. Robertson, I.D. Reid, J.M. Brady, What are you looking at? gaze estimation in medium-scale images, in: Proc. 16th British Machine Vision Conference (BMVC2005).

[21] B. Benfold, I. Reid, Unsupervised learning of a scene-specific coarse gaze estimator, in: Proc. the 13th IEEE International Conference on Computer Vision (ICCV2011), pp. 2344 –2351.

[22] C. Chen, J.-M. Odobez, We are not contortionists: coupled adaptive learning for head and body orientation estimation in surveillance video, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR2012), pp. 1544–1551.

[23] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME – J. Basic Eng. 82 (1960) 35–45.

[24] V. Prisacariu, I. Reid, fastHOG – A Real-Time GPU Implementation of HOG, Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009.

[25] C. Tomasi, T. Kanade, Detection and Tracking of Point Features, Technical Report, CMU-CS-91-132, Carnegie Mellon University, 1991.

[26] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proc. 7th International Joint Conference on Artificial intelligence, vol. 2, pp. 674–679.

[27] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartogr. Int. J. Geogr. Inform. Geovisual. 10 (1973) 315–354.

[28] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.

[29] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[30] B. Benfold, I.D. Reid, Gaze direction dataset, 2009. <http://www.robots.ox.ac.uk/~lav/Research/Projects/2009bbenfold_headpose/project.html>.

[31] G. Bradski, The openCV library, Dr. Dobb's J. Softw. Tools (2000).

[32] M. Krinidis, N. Nikolaidis, I. Pitas, 3d Head pose estimation using support vector machines and physics-based deformable surfaces, in: Proc. 9th International Symposium on Signal Processing and Its Applications (ISSPA2007), pp. 1–4.