修　士　論　文

Head Pose Estimation from Low Resolution Image with Scene Adaptation

(環境への自動適応を伴う低解像度画像からの頭部姿勢推定)

東京大学大学院
情報理工学系研究科
電子情報学専攻

48-106416　ジャムウェイハー　イサラン

指導教員　佐藤 洋一　教授

Abstract

Head pose estimation technique is a core of many computer vision applications. Head pose estimation is often used as cues to estimate human attention. Due to limitations such as camera placement location or system resources, it is not always appropriate to install high resolution cameras on every system. Head pose estimation from low resolution images are desired in such situation.

Although many methods have been proposed for head pose estimation from low resolution images, there are many technical limitations for the implementation of the system. To construct an effective head pose estimation system, a large number of head pose training samples collected from the same environment as test environment are desired because the process of acquiring such dataset is extremely time-consuming and makes it impossible to prepare scene-specific dataset for every scene.

The first part of this paper describes the method to automatically obtain scene-specific dataset. This method exploits the observation that people are more likely to to turn their head to where they are walking. With this observation, the tracking method is applied to the video taken from the scene beforehand. Head pose training data are then inferred from tracking results and are acquired automatically. This method enables automatic acquisition of training data and solves the problem of data collection.

The second part describes the method to improve head pose estimation in scenes with short available videos or low number of walking pedestrians so adequate amount of head pose samples cannot be captured. Datasets captured from various scenes are used to alleviate the problem and increase the accuracy significantly.

The third part describes the method which improves head pose estimation accuracy for scenes with dramatic difference within the scene. This method adaptively divides scenes into multiple parts in order to localize head pose estimator.

Acknowledgements

First of all, I would like to express my sincere gratitude to my professor Yoichi Sato for not only his excellent advices and his encouragements for the research but also his kindness for helping me to get used to living in Japan and have wonderful experiences here in Japan. I also would like to express my special thanks to Yusuke Sugano and Takahiro Okabe, who helped me a lot in my research and also for their advices regarding writing academic papers. If it were not for them, my research would not have been as successful as it currently is.

My thanks also go to my lab mates for their encouragements, advices and also their energetic personalities make me feel motivated to continue my research. My gratitudes also go to my Thai friends here in Japan, they not only make me feel like home but also their helpful advices help me get through a lot of problems in Japan.

I was fortunate that I was selected by Panasonic Scholarship. Not only the scholarship provides monetary aids and enables me to study here in Japan but also its numerous supports throughout my study here in Japan. Panasonic Scholarship's staffs extensively support my life in Japan and ensure that I live here without any problems. They always ask me whether I have any problems getting used to living in Japan or if I need any assistance. It is also because they encourage me to study Japanese language that I am able to live my life smoothly in Japan because Japanese language is very useful to living the life in Japan.

I am also grateful to Sato laboratory's secretaries, Yoko Imagawa, Sakie Suzuki and Usui Chio who make my life here in Japan smooth. They helped me from the start even before I come to Japan in arranging my Visa documents, tuition fee payment, dormitory applications and many more procedures.

Finally, my deepest thanks go to my family, my father, and my mother for their love, their support throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Head pose conveys a lot of information, for example, human head pose could convey where his attention point is. Moreover, human will turn their head to the person they are talking to as the nonverbal sign to direct his attention and to inform him that he is about to talk. As humans turn their head to the direction of the objects they are focusing on, head pose is an important factor in inferring the focus of attention of humans in computer vision.

Recently, estimating humans' visual focus of attentions has become very popular research trends as these researches enable a lot of applications in our daily life. One example would be to estimate attention of people in the meeting, making it possible to create automatic system to capture and understand the meeting flow. They could also be used with the camera fixed at the board to estimate human attention, to find out the poster people pay most attention to. For this reason, techniques for estimating head pose have been considered an important research task.

Although various image-based approaches have been proposed for estimating head pose (see [24] for a recent survey), one of the major remaining technical challenges is to deal with low resolution images. In some application scenarios like visual surveillance, it is often the case that head regions in input images are quite small. Small images contain limited information, thus it is still a challenging task to achieve accurate estimation results in such cases.

Recently it has become well known that the use of appearance-based approaches is a promising way to estimate head poses from low resolution images. Compared with model-based methods like active appearance models [11, 22] which rely on geometric facial models, appearance-based methods

directly treat image features and are known to work even with low resolution images.

Accuracy of the appearance-based head pose estimation method relies heavily on a good dataset. Systems trained with dataset containing samples similar to test samples can achieve high efficiency while dataset different from the scene yield poor results when used. Therefore, a dataset is usually taken from the same scene as target scene. Such dataset is called scene-specific dataset. However, there is currently no publicly available datasets for head pose estimation on low resolution images. The ground truth dataset for each work is mostly taken from low resolution video and hand-labeled the direction.

Orozco *et al.*[25] use manually cropped 800 head images, 100 for each pose class from i-LIDS[17] dataset. Gourier *et al.*[16] use downsampled images from Pointing'04 dataset into low resolution image of dimension 23x30. In Robertson *et al.*[29][30], ground truth has been produced by a human user drawing the line-of-sight on the images.

To obtain ground truth labels, training data need to be manually labeled or an intrusive device is needed to record head pose directions. More importantly, head appearances can change significantly from scene to scene, and according to camera properties even in the same scene. Accordingly, head pose estimators work best if trained with data from the same camera and setting. However, it is prohibitively expensive to collect ground truth data manually every time a head pose estimation method is applied to different scenes.

In some cases, there are not always enough head pose training data for training estimator models. Estimators trained with those data would not be generic enough to cover all possible head pose appearances. This problem can be alleviated by introducing generic dataset, which were taken from other domains, to help training estimators. Although, as mentioned before, high accuracy could not be expected from generic dataset, with the technique called transfer learning which has recently being extensively researched, generic data could be made use of in order to improve generalization of the estimator while retaining the accuracy for the domain.

Transfer learning is the technique used in cases where there are not enough training data which were taken from the same domain as test data, these data are called *target data*. The problem can be alleviated by introducing the data called *source data* which is different from target data but is related in some ways to the target data. Although source data is not collected from the same domain, the abundant amount of the data can be made use of by the transfer learning techniques to improve estimation accuracy of the estimators. For more information, we refer readers to the extensive survey made by Sinno *et*

al. [26].

When labeled data in both domains are available, transfer learning can improve estimation accuracy by using knowledge from source domain to reinforce the estimator for target domain. For example Kamishima *et al*. [19] performs transfer learning together with bagging technique to combine weak learners to create a strong learner which improve performance for estimation tasks. Transfer learning has shown to be successful in data mining and machine learning tasks, however, there has not been any works which apply transfer learning techniques with head pose estimation from low resolution images yet.

Furthermore, even in the same scene, there could be severe head pose appearance difference. Some caused by severe illumination differences. For example some areas might reside in building shadow while the others in direct sunlight. Camera parameters, especially camera angles could also cause appearances in the same scene to be different. Appearance difference might cause head pose estimators trained with these data to be inaccurate due to large variations of head pose samples from the same class. However, no work as of now has yet solve this problem.

From this section, it can be seen that there are a lot of problems yet to be solved and by offering our solutions to the problems we aim to contribute to the computer vision society and hope to enrich research activities in this area.

1.2 Overview

In this section, we gave an overview of our proposed solutions to existing problems on head pose estimation from low resolution images.

In Chapter 3, the method that automatically collects training dataset from test scenes is described. Based on the observation that people tends to turn their head towards where are walking, we track pedestrians in the video in order to automatically acquire head pose training samples Pedestrians in the input image sequence are tracked first to achieve their head images and walking directions. After rejecting outliers which are facing different directions, their walking directions are used as ground truth labels of their head orientations. In this way, our method does not require a tedious and time-consuming task of collecting a large amount of ground truth data. Figure 1.1 shows the framework of our proposed method.

In Chapter 4, we describe the method to solve the problem of highly biased distribution of training samples. The idea is to integrate a generic dataset, *i.e*., manually labeled head images taken from a different scene, with

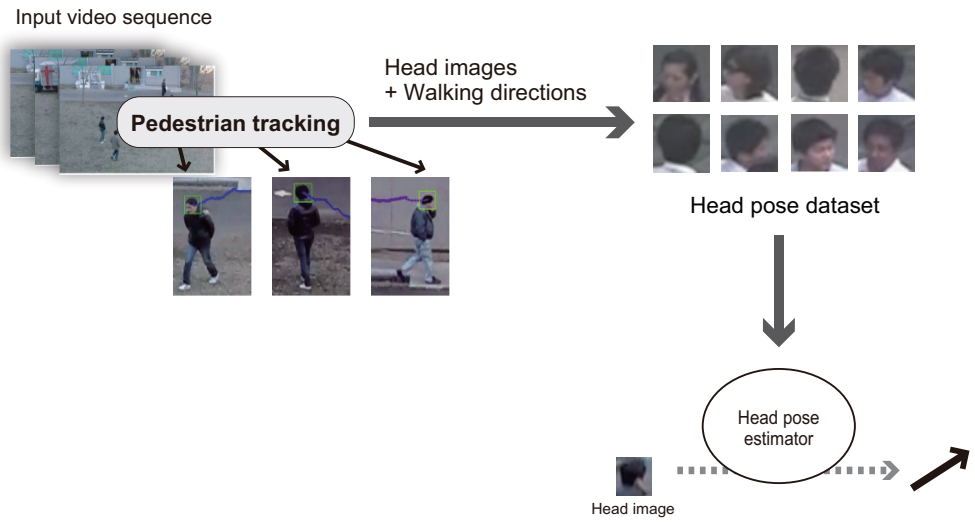


Figure 1.1: Proposed framework for chapter 3. Given an input video sequence, our method first track pedestrians in the video and obtain their head images and direction they are walking. By using the walking directions as a cue to infer head pose directions, out method constructs an appearance-based head pose estimator.

the automatically generated scene-specific dataset. Higher accuracy cannot be expected for generic datasets as discussed above, however, to complement the drawbacks of both datasets, our method seamlessly integrates the scene-specific dataset and a generic dataset via transfer learning technique. Figure 1.2 shows the framework of our proposed method.

Chapter 5 describes the method which aims to solve the problem of appearance differences within the same scene. The scene is divided into multiple regions using graph segmentation method and a classifier is constructed separately for each region. Figure 1.3 shows the framework of our proposed method.

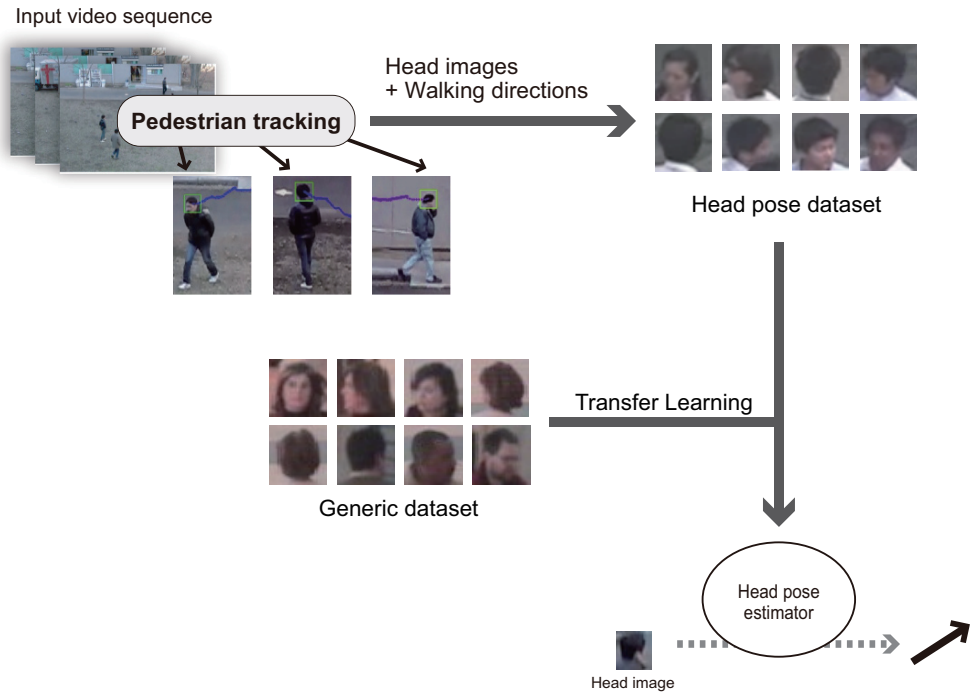


Figure 1.2: Proposed framework for chapter 4. Generic dataset is integrated into the model by transfer learning technique.

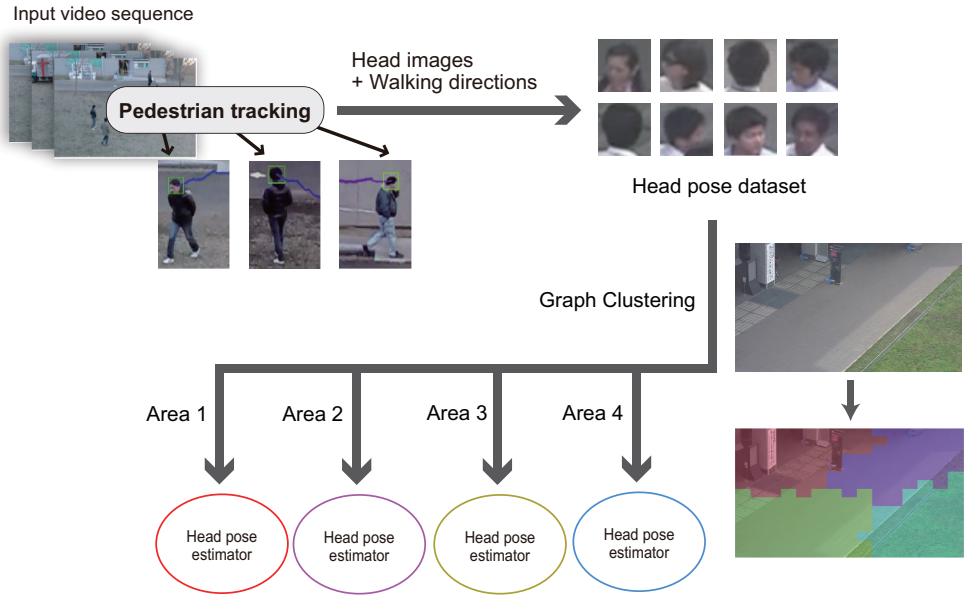


Figure 1.3: Proposed framework for chapter 5. A scene is intuitively divided into multiple regions with similar head appearances and classifiers are constructed separately for each area.

Chapter 2

Related Works

Recently there have been numerous attempts to tackle the problems in head pose estimation from low resolution images. Due to the difficulty of this problem, this field is being researched by many research groups and is being paid a lot of attentions to in the field of computer vision.

Referring to the categorization by Murphy *et al.*[24]'s survey on head pose estimation, recent approaches for head pose estimation in low resolution images can be categorized as follows.

2.1 Appearance-Based Method

Appearance-Based method use image-based comparison method to match the input with the examples from each pose class and select the pose which match the input the best as the estimation. Usually, the descriptor is extracted from the image for comparison. Finding the good descriptor for low resolution image is still a challenging task for appearance-based method. Then the descriptor is then divided into multiple classes using many techniques such as binary search tree[29], Randomized fern[6], multi-class Support Vector Machine (SVM)[25]

Robertson *et al.* [29] used skin color as a descriptor and a binary tree algorithm to establish a head pose classifier. Body direction is also used to filter out poses which is not physically available such as human walking to the north direction but his head turns to south. Figure 2.1 shows the example of the feature descriptor and estimation results of this work.

Benfold *et al.* [5] proposed a descriptor which learns a model of skin color automatically and used a randomized fern algorithm for head pose classification. Figure 2.2 shows the example of the feature descriptor and estimation results of this work.

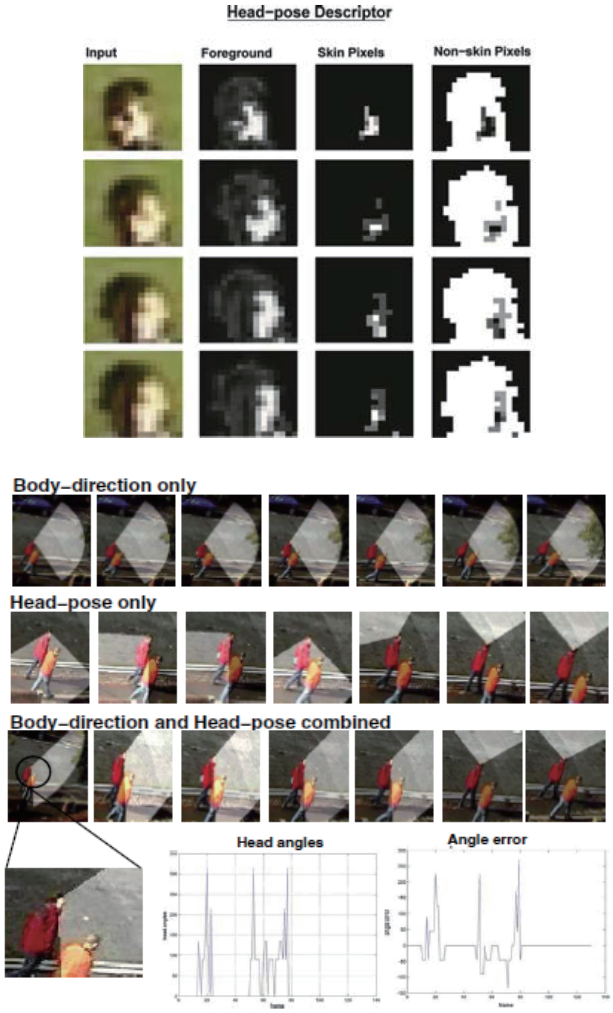


Figure 2.1: Example of the feature descriptor and estimation results of the method from Robertson *et al.* [29].

Orozco *et al.* [25] proposed an image descriptor which does not require explicit segmentation of skin and hair pixels by using similarity distance maps with class-mean appearance templates, and used the descriptor with a multi-class SVM (support vector machine) for head pose classification. Figure 2.3 shows the example of the feature descriptor and estimation results of this work.

Their work has been applied to surveillance videos, and it is shown that head poses can be estimated even from low-resolution head images.

However, current appearance-based methods suffer from one important

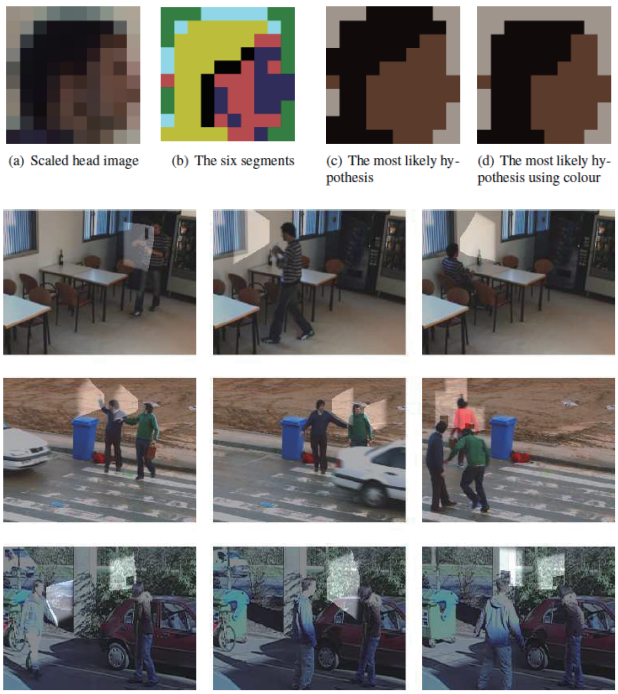


Figure 2.2: Example of the feature descriptor and estimation results of the method from Benfold *et al.* [5].

problem when they are used in realistic scenarios. That is, a large number of training images with ground truth labels, *i.e.*, correct head orientations, are needed. For instance, Orozco *et al.* [25] and Robertson *et al.* [29, 30] used 100 images for each head pose class as training data.

To alleviate this problem, Benfold *et al.* [7] uses tracking method to track and automatically capture head pose images and its tentative directions and uses Conditional Random Field to model the interactions between the head motion, walking direction, and appearance to recover the gaze directions. Figure 2.4 shows the example of captured head images and Conditional Random Field's factor graph of this method.

The methods in this category have the advantage in the ability to train the model using only positive samples without the need of negative samples. Furthermore, expanding the samples to the template model could be done anytime. These methods work well with very low resolution head pose images for head size as small as 10 by 10 pixels.

However, these methods usually require good localization of the head image and localization of low resolution head images is not a trivial task,

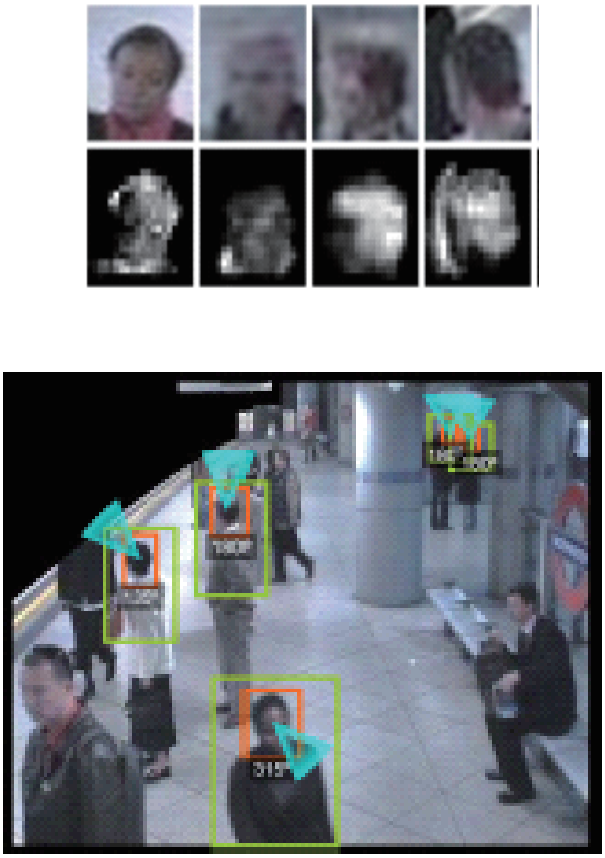


Figure 2.3: Example of the feature descriptor and estimation results of the method from Orozco *et al.* [25].

therefore good localization techniques are needed for these methods.

2.2 Detector Arrays Method

Detector arrays methods use techniques similar to face detection, which has been developed and was very successful in the past. The method use separate face detectors for each pose class, then assign the pose with the greatest score from the detector.

Zhang et al.[39] uses FloatBoost classifiers, which is a variant of AdaBoost[36], to classify head poses into 5 classes.

These methods have the advantage that they do not require head localization techniques because the detectors are trained to detect the head for each

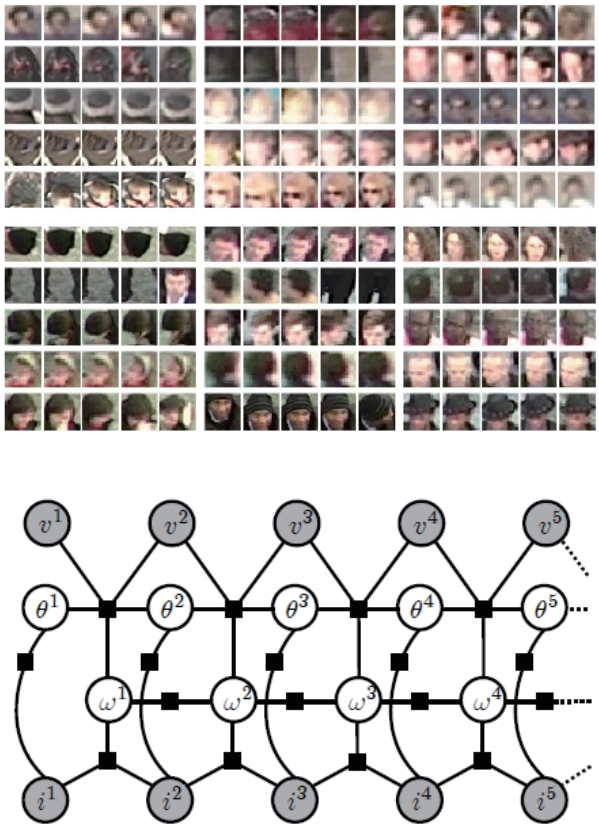


Figure 2.4: Example of the captured head images and Conditional Ramdom Field's factor graph in Benfold *et al*. [7].

pose. However, training the detectors is burdensome, because they require a lot of training data, including both positive and negative ones.

The problem also arises when multiple detectors simultaneously detect the same window; it would be hard to determine what pose the input should be assigned to. Rowley et al.[31] suggests the solution using router method, which first assume that the window contains the face and determine its orientation, then rotate the face and use the face detector to confirm the existence of the face. So the detector can be trained with only small range of rotation, in this work, the detector is trained within the range -10° to +10°

2.3 Nonlinear Regression Method

Nonlinear regression methods map the head pose from the input image space to the pose directions. Usually the training data is labeled with discrete or continuous angles. Examples pf the nonlinear regression methods are neural network and Gaussian process regression methods.

Stiefelhagen et al.[34][33] uses separate neural networks to train and classifies the pan and tilt angles of the head pose. Gourier et al.[16] uses linear auto-associative neural networks techniques to estimate the head pose.

These approaches have the advantage of being fast due to it only requires positive samples, but this method also requires good localization of the head and is prone to errors from bad localization.

2.4 Tracking methods

Tracking methods utilize the movement between frames of the head images to estimate the head pose. These methods usually require proper initialization of the head image.

Pappu et al.[27] generates the synthetic views offline using head tracking methods, then find candidate head poses from the generated views and find the pose which minimizes the difference in appearance between the input and the generated head poses.

Morency et al.[23] uses stereo-motion based head tracking which utilizes depth and brightness gradient tracking combined with initialization and stabilization modules to produce system robust to strong illumination changes.

Wu et al.[38] builds ellipsoidal model of points, where each points maintain probability density functions of local image features and use maximum a posteriori (MAP) estimation to estimate the pose. This technics is robust to illumination change due to the use of local edge density feature which is independent of person and illumination.

These methods have the advantages that the results are quite accurate due to the ability to track small drifts between images. However, these methods require first initialization of the head to track, which means these approaches requires the face image to be in some pose in order to initialize the system and the system could not handle large drifts between image frames, making the application unusable in some systems.

It can be seen that each approach has its own advantages and disadvantages. What method to use for each situation will depend on contexts of the scene such as image resolution, image appearances and number of data available.

As our task aims to determine head pose from images as low resolution as 10 by 10 pixels, from all approaches mentioned, appearance-based head pose estimation methods are the most suitable ones and become the focus of our approach.

Chapter 3

Head Pose Estimation with Automatic Data Acquisition

This chapter describes the appearance-based head pose estimation method which address the problem that it is extremely difficult to obtain head pose training data for every scene.

3.1 Appearance-Based Head Pose Estimation

Head Pose Estimation is a task to determine head pose of the given image. Head pose estimation ranged from coarsest level, which head poses are divided into multiple discrete classes, to the finest level, which head poses are estimated across multiple degree of freedoms (DOFs).

In low resolution images, however, the information on head poses are usually not enough to infer head pose up to multiple degrees of freedoms and thus it is generally accepted that it is enough to conduct head pose estimation on head pose direction relative to the image. Head pose estimation tasks from low resolution images are usually divided into 2 categories, classification task and regression task. Figure 3.1 shows an example of general head pose estimation task from low resolution images. In classification task, head poses are divided into multiple discrete classes. In regression task, head poses take continuous values between 0 and 2π.

Appearance-based head pose estimation methods are shown to be the most successful method for head pose estimation from low resolution image. These methods assume that head regions have been detected and correctly localized, thus robust head detection and localization are required for appearance-based head pose estimation methods.

Appearance-based head pose estimation methods are usually divided into

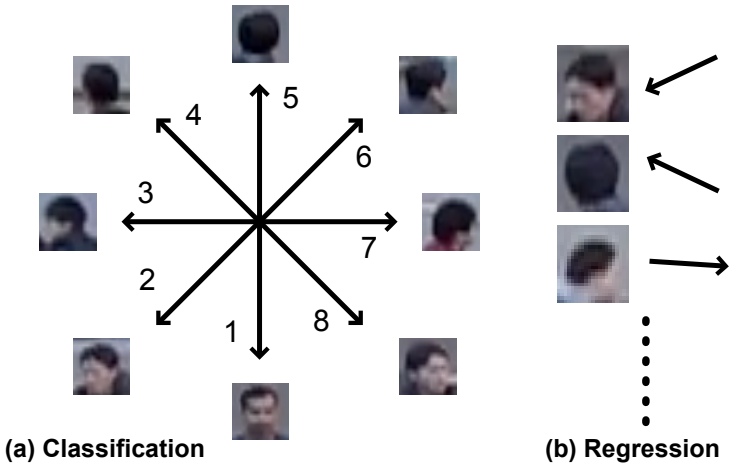


Figure 3.1: Illustrations of head pose estimation tasks. Head pose is defined as (a) discrete classes in the case of classification and (b) continuous values between 0 and 2π in the case of regression.

3 phases, data collection phase, training phase and pose estimation phase. Figure 3.2 shows the summary of these 3 phases.

In data collection phase, head pose labels are collected. Head pose label describe head pose appearances of each pose class and will be used for training the model. Each head pose label $\boldsymbol{l} = \{\boldsymbol{I}, \theta\}$ consists of head image $\boldsymbol{I}$ and its corresponding pose direction θ.

Because image intensities might not effectively convey information on the head pose, new representation of the image called feature descriptors are extracted instead. Feature descriptors are a set of values which describe the image in a way which is suitable for the task. For high resolution image, feature descriptors might be the position of the eyes, mouth and nose of the person. In the low resolution case, however, those features could not be reliably detected so usually more low level features such as Histogram of Oriented Gradients (HOGs) [12] or horizontal and vertical edges are extracted. In appearance-based head pose estimation, feature descriptors $\boldsymbol{h}$ are extracted from each head pose image and a training dataset $D = \{\boldsymbol{p}\}$ containing training samples $\boldsymbol{p} = \{\boldsymbol{h}, \theta\}$ is constructed.

This dataset is very crucial to the accuracy of the system. If the prepared head pose images are similar to the test dataset, high accuracy of head pose estimation could be expected. On the other hand, low estimation accuracy occurs if prepared head images are different from test data. Therefore, the training datasets are usually collected from the same scene with the same

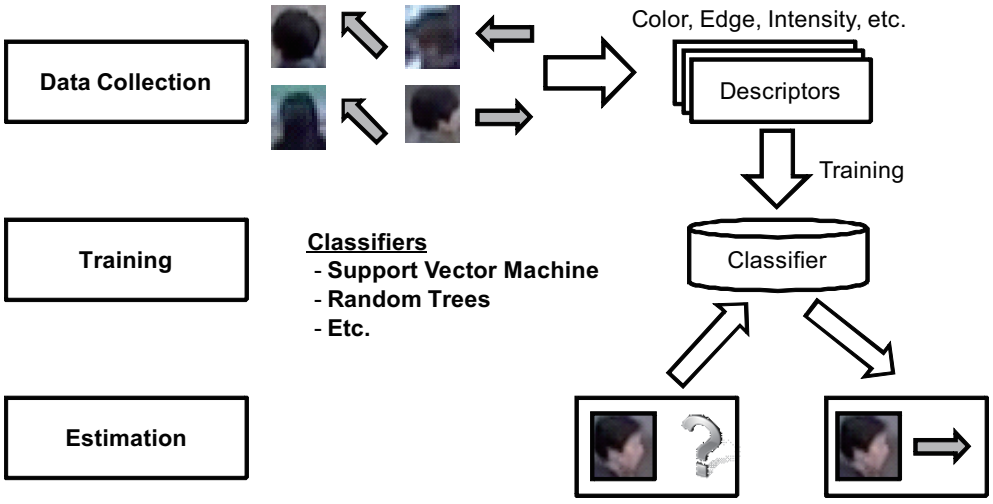


Figure 3.2: Appearance-based head pose estimation methods are usually divided into 3 phases, data collection, training and pose estimation phase.

setting as test data.

Estimator models for head pose estimation from low resolution images can be divided into classifiers for classification tasks, and regressors for regression tasks. With the input feature descriptor $\boldsymbol{h}$, regression models $\hat{\theta} = f(\boldsymbol{h})$ output the estimated value $\hat{\theta}$. Classification models $p(\theta) = f(\boldsymbol{h}), \theta \in \boldsymbol{\theta}$ output the probability $p(\theta)$ for each class in all available classes $\boldsymbol{\Theta}$. In the training phase, given head pose training samples $p = \{\boldsymbol{h}, \theta\}$ with feature descriptors $\boldsymbol{h}$ extracted from images $\boldsymbol{I}$ in the training dataset D is used to estimate the mapping function f.

Head pose estimator is usually one of the machine learning estimators such as support vector machines (SVM), random forest [9] or randomized ferns. The choice of the estimator is one of the most important factors in head pose estimation task. Estimators suitable for the task at hand will yield more estimation accuracy than other estimators.

Finally, head image with unknown pose is taken as input to the model. For regression task, the continuous value $\hat{\theta}$ is estimated. For classification task, the probability $p(\theta)$ of each class is estimated. In the simplest form of application, the class with the highest probability is chosen as the estimated head pose and the task could also be written in the same manner as regression model $\hat{\theta} = f(\boldsymbol{h})$.

As discussed earlier, an important problem that was largely ignored in the previous studies on head pose estimation from low resolution images is how to obtain appropriate training samples. Since we assume the under-

lying mapping function $f(\boldsymbol{h})$ is identical in both training and test scenes, classification accuracy highly depends on how similar these training and test scenes are. In other words, if lighting conditions and camera positions are significantly different between the scenes where training and test images are taken, mappings between pose and appearance would also become different. However, it is not always possible to collect training samples for every test case.

3.2 Proposed Method

Our basic idea is to use walking direction as a cue to acquire training samples with automatically assigned labels of their head poses. Figure 1.1 shows a basic framework of our method. Given an input video sequence, we first track pedestrians in the video and obtain their head images and directions they are walking in. As these pedestrians are most likely to turn their head to their walking directions, the walking directions can be assumed to indicate the head poses of the images.

However, this idea cannot be applied in a straightforward manner. Since people can move their heads freely even while they are walking, it is obvious that our basic assumption does not always hold and the training labels contain a certain amount of noise. Head pose estimation algorithms are not always robust to such outliers, and thus it is ideal to reject them prior to the learning stage. Furthermore, walking directions are unevenly distributed in most of the scenes, and this can result in a biased estimation result with larger error.

To address this problem, we introduce a strategy to reject unreliable data from the tracking results. Each tracking trajectory is first divided into straight line segments in which each pedestrian walks in a straight line. Unreliable line segments are rejected and then one representative image per line segment is constructed and used as the training data. Oversampling is then applied to handle the imbalanced dataset. Details of the proposed strategies are described in the following sections.

3.2.1 Pedestrian Tracking

We used the Benfold *et al.*'s method [6] to track pedestrians in input videos. The method combines a head detector and velocity estimation with feature tracking. With this method, not only are the head of people in the video properly tracked but method also yields good results on centering the head, which is in most cases vital to appearance-based head pose estimation. Here

the pedestrian tracking method is briefly explained for reader's benefit. For more details, readers are referred to [6].

The tracking method is based on a Kalman Filter [18] with two types of measurements. The head location from the head detector based on a histogram of oriented gradients (HOG) [28] and head movement velocity by combining the velocities of multiple tracked corner features [35, 21]. For every tracked frame, the head image $\boldsymbol{h}$, the head location $\boldsymbol{u} = (x, y)$, and the size of covariance matrix for each location measurement $\boldsymbol{c} = (c^{(x)}, c^{(y)})$ that can be used as an error measurement is then collected for analysis.

The pedestrian tracking algorithm is applied to the whole input sequence and a trajectory, *i.e.*, a set of head images $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}$, head locations $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}$ with error measurements $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$, is acquired for each pedestrian. N denotes the length of the trajectory and it varies for each trajectory.

3.2.2 Walking Direction Estimation

As discussed above, our method first divides the trajectories into straight line segments. More specifically, each trajectory is divided into M segments $\{S_1, \ldots, S_M\}$ by polyline simplification using the Douglas-Peucker algorithm [14]. Douglas-Peucker constructs a minimal set of lines so that the orthogonal distance from each point to the nearest line is less than a threshold. The algorithm starts by taking a set of points $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}$, then constructs a line from point $\boldsymbol{u}_1$ to $\boldsymbol{u}_N$. The algorithm then finds a point $\boldsymbol{u}_n$ with maximum orthogonal distance from the line. If the distance is more than a threshold, the algorithm divides the point set to $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ and $\{\boldsymbol{u}_n, \ldots, \boldsymbol{u}_N\}$ and repeats the above process on these two sets recursively. The algorithm stops when the maximum distance becomes less than the threshold.

Figure 3.3 shows an example of polyline simplification. It can be seen that the pedestrian in this image is not walking straightly, and thus treating all images as one direction will definitely be erroneous. With the polyline simplification algorithm, the trajectory is divided into 4 line segments in which the pedestrian walks straight. In the figure, the curved line shows the raw tracking result and straight lines show line segments obtained using the polyline simplification algorithm.

Next, the walking direction of each line segment is estimated. Since polyline simplification only considers the start and the end point of each segment, using polyline simplification to estimate the pose direction may yield an inaccurate result. Therefore, a line fitting method which considers all points in the segment is employed to analyze and estimate the pose direction

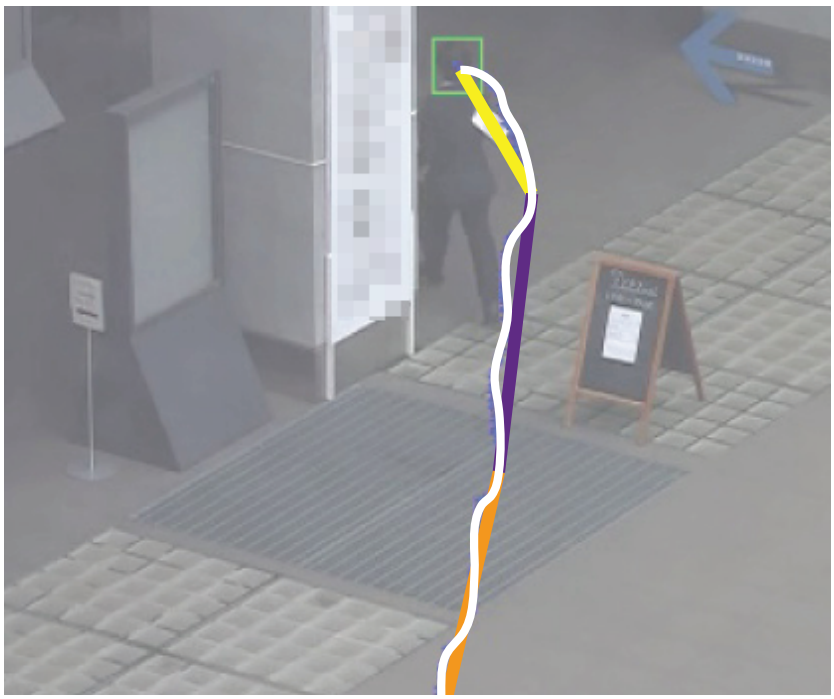


Figure 3.3: Example of polyline simplification result. Curved line shows tracking result and straight lines show simplification result.

for each segment. Given a set of T tracked head locations in the segment, a line which minimizes a sum of residuals is computed and the direction of the line θ is assigned to the segment as the walking direction.

3.2.3 Outlier Segment Rejection

After the polyline simplification and the line fitting, walking directions can be estimated accurately for pedestrians and can be used as their head orientations. However, as discussed above, walking directions do not always correspond to head orientations and line segments are not always suitable for training samples and a scheme for rejecting outlier segments is necessary.

In this work, we apply four rules to reject segments: 1) with a high number of erroneous points, 2) in which a person walks short distances or walks slowly, 3) with large line fitting errors, and 4) with high image variance. The details of each rule are as follows.

Segments with a high number of erroneous points Let us denote the T head locations in the segment as $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\}$. Segments with many

erroneous points, *i.e.*, points that are regarded to be false positives of the tracking algorithm, are rejected because of low reliability. Specifically, a point $\boldsymbol{u}_t$ is judged as erroneous if the error measurement of the tracker is significantly large compared to head sizes:

$$\frac{c_t^{(x)}}{s_x(\boldsymbol{u}_t)} > \alpha \text{ and } \frac{c_t^{(y)}}{s_y(\boldsymbol{u}_t)} > \alpha. \tag{3.1}$$

where $\boldsymbol{c}_t = (c_t^{(x)}, c_t^{(y)})$ is the error measurement of the corresponding frame and α is a constant value. $s_x(\boldsymbol{u})$ and $s_y(\boldsymbol{u})$ are position-dependent head width and height defined as:

$$s_x(\boldsymbol{u}_t) = Ax_t + By_t + C \text{ and } s_y(\boldsymbol{u}_t) = Rs_x(\boldsymbol{u}_t), \tag{3.2}$$

which assumes that heads are fixed size and moving on a plane under a perspective projection. The parameters A, B, C and R are manually set. Using this measure, reject segments if the number of erroneous segment points is larger than a predefined threshold τ_e.

Segments with short distance or slow movement Short segments are better to be rejected since they do not have enough information for the line fitting. Similarly, segments with slow walking speed are rejected since the pedestrians are likely to be doing something else, *e.g.*, talking with each other and not facing straight. Specifically, reject segment if

$$\frac{|\boldsymbol{u}_T - \boldsymbol{u}_1|}{\bar{s}} \leq \tau_n \text{ or } \frac{|\boldsymbol{u}_T - \boldsymbol{u}_1|}{T \cdot \bar{s}} \leq \tau_v \tag{3.3}$$

where τ_n and τ_v are predefined thresholds. $\bar{s} = \sum_{t=1}^{T} \sqrt{s_x(\boldsymbol{u}_t)^2 + s_y(\boldsymbol{u}_t)^2}/T$ is the average head size factor of the segment and introduced to make the measurement scale-invariant.

Segments with large line fitting error If the line fitting error is large, it is natural to assume that the person is moving in a curve or the tracker failed to track the head. Therefore, reject segments if

$$\sum_{t=1}^{T} \frac{|y_t - g(x_t)|}{\sqrt{m^2+1} \cdot |\boldsymbol{u}_T - \boldsymbol{u}_1|} \geq \tau_l, \tag{3.4}$$

where τ_l is a threshold and the left side of the equation is a scale-independent line fitting error of the estimated line $y = g(x) = mx + c$.

Segments with high image variance Segments with high image variance imply unstable head images due to, *e.g.*, frequent changes of head orientations. Specifically, we calculate the variance of resized I-dimensional head image vectors $\{\hat{\boldsymbol{h}}\}$. Segments are considered to have high variance if

$$\frac{\sum_{t=1}^{T} |\hat{\boldsymbol{h}}_t - \bar{\boldsymbol{h}}|^2}{T \cdot I} \geq \tau_{var}, \tag{3.5}$$

where $\hat{\boldsymbol{h}}_t$ denotes the t-th resized image, and $\bar{\boldsymbol{h}}$ is a mean image calculated from all resized images $\hat{\boldsymbol{h}}$ in the segment.

3.2.4 Selecting Representative Images

With the rules above, most outlier segments are rejected and the remaining segments contain correct data. One representative image per segment is then selected and used as training data. Since only one orientation is assigned to each segment, most of the images in the accepted segments are redundant. Moreover, it is beneficial to use only one image per a segment in order to reduce computational cost of training classifiers.

In this work, we propose and examine three different selection methods. Basically, we select the image which 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 $\hat{\boldsymbol{h}}_t$ in the segment and the image with lowest distance is selected. This enables us to select the most representative image without suffering from effects that can be seen in the mean image, *e.g.*, blur or distortion. However, it is not always the case that blur and distortion cause poor estimation results. We also found simply using the mean or median image as the representative image can be an option in some cases. Further discussion will be given in Section 3.3.

3.2.5 Handling Imbalanced Data by Oversampling

By applying these processes to all of the successfully tracked pedestrians, a scene-specific dataset is acquired. Figure 3.4 shows an example of a distribution of walking directions in the sequence 1 (See section 3.3 for more details). In this sequence, the majority of the pedestrians walk in down-left (class 2) and up-right (class 6) directions. Imbalanced data causes low accuracy of head pose estimation for directions with few training samples. In order to handle this problem, oversampling is used with our classifiers. Oversampling technique is proved to be beneficial in reducing the effect of imbalanced data for classification task [37, 3]. The oversampling technique

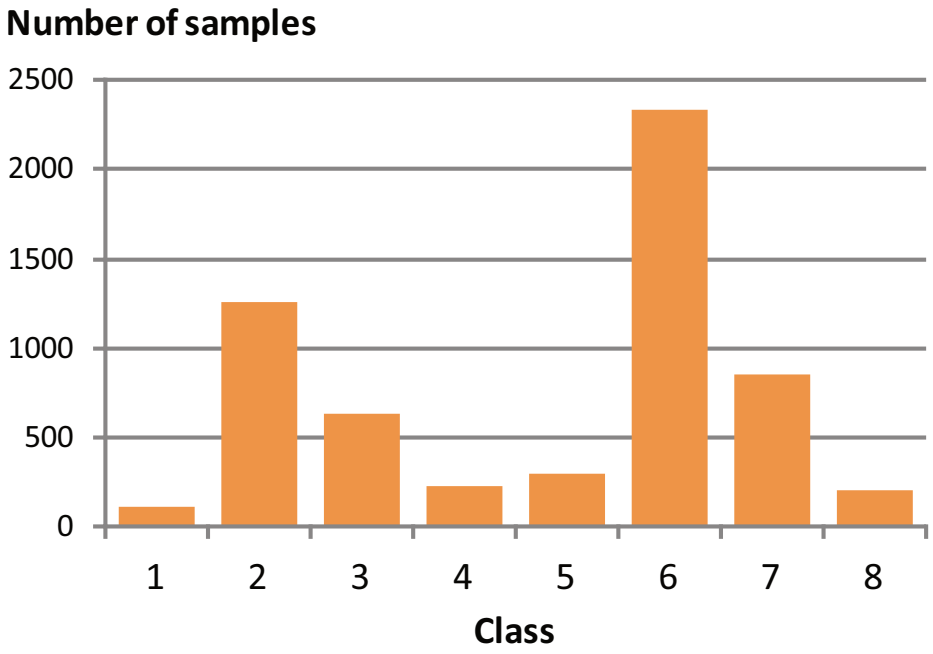


Figure 3.4: Head pose frequency captured from the video sequence 1. The horizontal axis indicates tracked walking directions and the vertical axis indicates numbers of samples obtained from the video sequence. Each class number is defined in same way as in Figure 3.1.

resamples data from classes with small amounts of data until every class has an equal number of data. By using oversampling, the accuracy of the classifiers is significantly improved.

3.3 Experimental Results

In this section, we present experimental results to demonstrate the effectiveness of our method. As mentioned earlier, it is hard to collect ground truth data manually for every scene, thus we show that training data from our method which automatically generates scene-specific training data performs better than using available data from other scenes. In order to demonstrate that our method is not limited to one classifier, a multi-class SVM classifier and a Random Trees classifier are also tested. The effectiveness of the head image selection method is also compared to other alternatives. The effect

Table 3.1: Settings of scene-dependent parameters.

Sequence	1	2	3
A	0	0	0.008
B	0.014	0.04	0.04
C	36.4	8	33
R	1.1	1.1	1.1
τ_n	3.0	0.5	0.5
τ_v	0.03	0.03	0.01
τ_{var}	0.0035	0.0055	0.0035

of imbalanced data and the effectiveness of oversampling method are also shown.

3.3.1 Experiment Settings

We conducted experiments using 3 video sequences with different length, which were recorded using different cameras in different scenes. Example frames in the video are shown in Figure 3.5. The scene images are input video frames with pedestrian tracking results overlaid. Examples of obtained head images are also shown with its estimated walking direction shown on their right part of the image.

The resolution of sequence 1 was 1920×1080 pixels and recorded at 30 fps for approximately 7 hours. Sequence 2 was 1120×780 pixels and recorded at 30 fps for approximately 10 minutes. Sequence 3 was 1280×720 pixels and recorded at 30 fps for approximately 10 minutes. As a result of pedestrian tracking, direction estimation and image selection, 5930 head images were captured from sequence 1, 1265 head images were captured from sequence 2, and 564 head images were captured from sequence 3. At the same time, test images (320 for sequence 1, 314 for sequence 2 and 227 for sequence 3) with manually-labeled ground-truth head poses were acquired from the same sequence and estimation accuracy was evaluated using these test images. To construct a generic dataset, 1477 samples were taken from Gaze Direction Dataset [4], which has been used in [6]. The set is divided by head pose into 8 classes and each class contains $100 \sim 200$ images. Figure 3.6 shows examples of head images included in the generic dataset.

In the experiments, the parameters were empirically set as follows; $\alpha = 1.0$, $\tau_l = 0.8$, $\tau_e = 0.4$, and other scene-dependent parameters were set as summarized in Table 3.1.

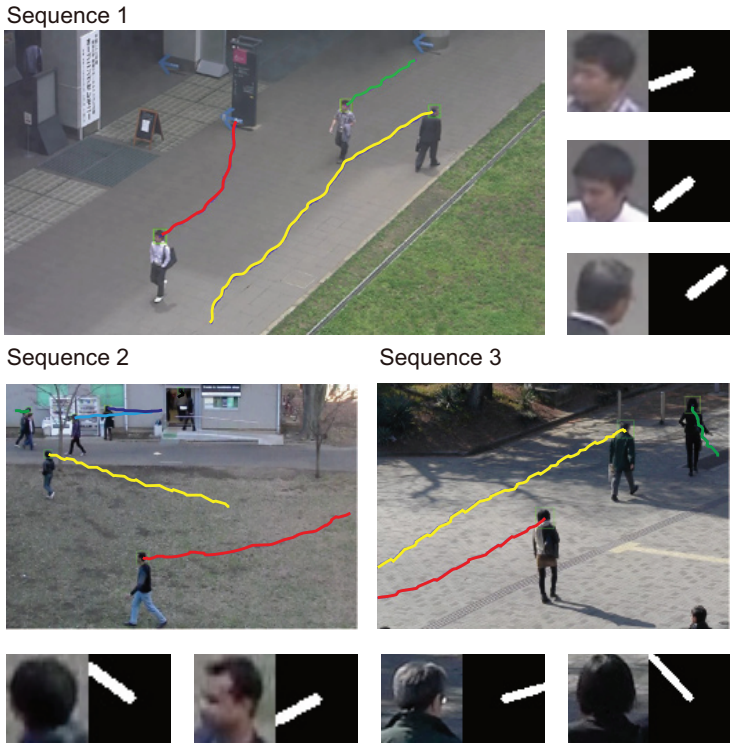


Figure 3.5: Example frames in the test video sequences. The input video frames are overlaid with pedestrian tracking results . Examples of obtained head images are also shown, the right part of each image presents its estimated walking direction.

3.3.2 Head Pose Estimation Test

For the head pose estimation method, we conducted two classification tests. The first classification test uses a linear SVM as classifier from the Liblinear library [15] and the second test uses a Random Trees classifier [10] from the OpenCV library [8]. All of the head images were converted to gray scale, normalized and resized to 20×22 pixels. Feature vector was defined as a 440-dimensional raster-scanned and normalized image vector in the following experiments. To evaluate the effectiveness of the proposed method, we compared two results: **Generic** result based on the generic dataset and **Proposed** result based on our method.

Classification accuracy comparison between our proposed method and the generic dataset is summarized in Figure 3.7. The accuracy is calculated from the average accuracy of each class. Standard deviations are indicated

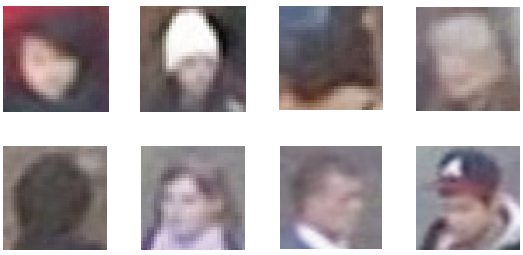


Figure 3.6: Example images in the Gaze Direction dataset.

as error bars.

As can be seen, the accuracy of classification using the generic dataset is significantly lower. In contrast, our proposed method achieved higher accuracy than the generic result. The result also shows that the accuracy improved a lot for sequence 1 which utilizes 7 hours video. It is also shown that even for only 10 minutes of video as in sequences 2 and 3, the accuracy is significantly improved over using the generic dataset. Standard deviations also become smaller in the proposed result.

We also compared our representative image selection method with 2 other selection methods. For the first alternative, we calculate the mean image from images in each segment and use it as the representative image. For the second alternative, we calculate the median image. Each pixel in the median image is constructed from the median pixel intensity at its location over all images in the segment.

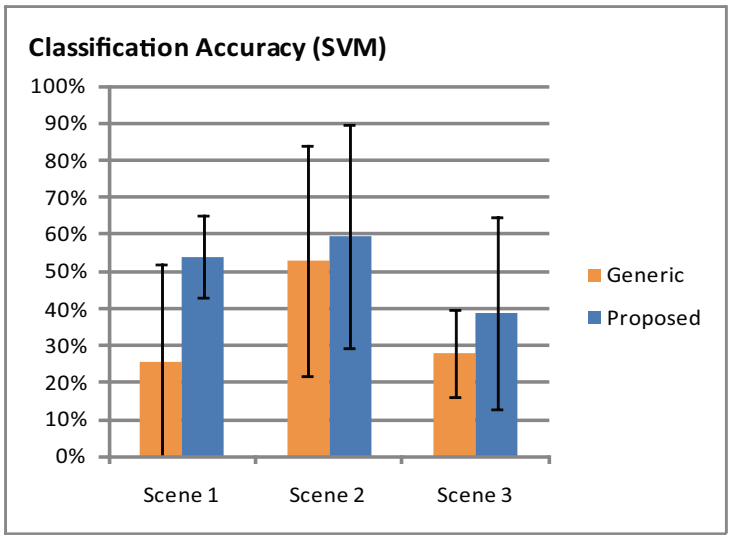
Classification accuracy for each image selection method is summarized in Figure 3.8. It can be seen that the Mahalanobis image selection method performs comparatively well to other methods. However, it should be also noted that the other two methods also show better results than the generic results in Figure 3.7. It indicates that our proposed idea has robustness to the image selection method.

Figure 4.2.1 shows confusion matrices of the classifiers. We compared confusion matrices of classifiers applied to test data from scene 1. The **Generic** result uses generic dataset as training data for the classifiers, the **Imbalanced** result uses data obtained from our method without oversampling data to train the classifiers, and the **Proposed** result uses data obtained from our method applied with oversampling to train the classifiers. The effects of imbalanced data can be seen in Figure 3.9(c) 3.9(d). As the majority of pedestrians in the sequence walked in the direction as shown in Figure 3.4, the results are biased towards class 2 and 6. The improvement using oversampling can clearly be seen in Figure 3.9(e) and 3.9(f).

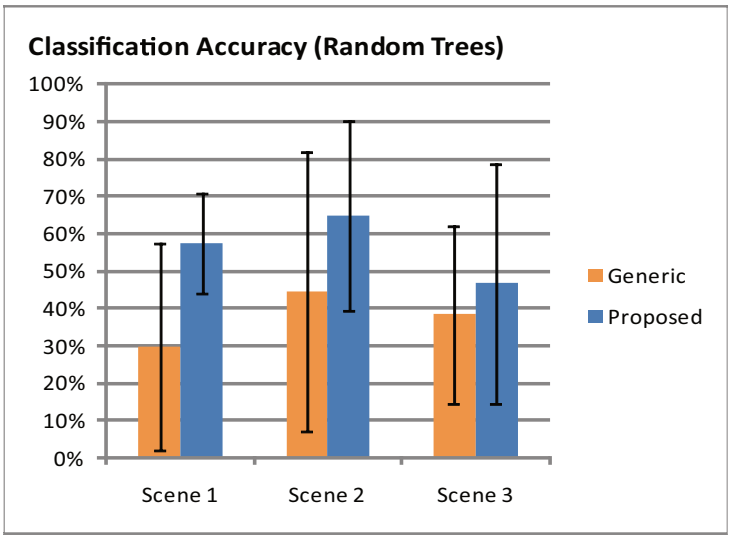
3.4 Conclusions

This chapter proposes the head pose estimation method with automatic scene-specific adaptation. We automatically acquire scene specific head pose samples without manual labeling. The result also shows that the accuracy of the classifier trained with data acquired with our proposed method improves significantly over the classifier trained with generic dataset.

With this method, head pose estimation system could be constructed without requiring extremely time consuming manual works as traditional methods.

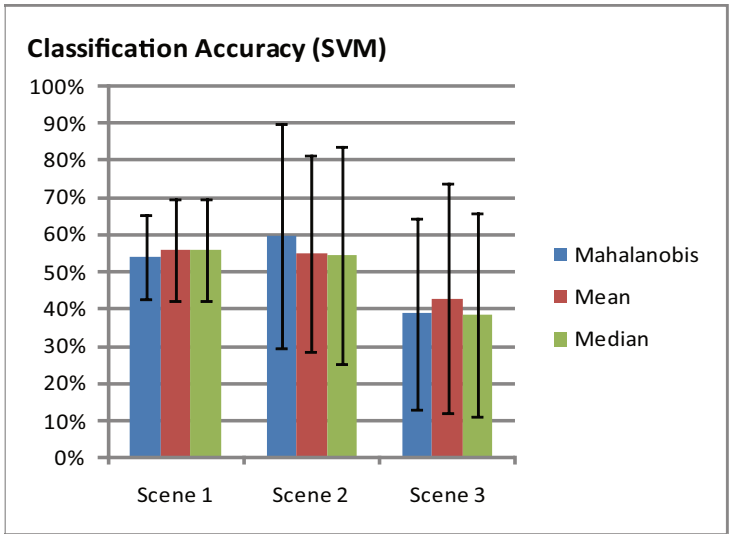


(a) SVM classifier

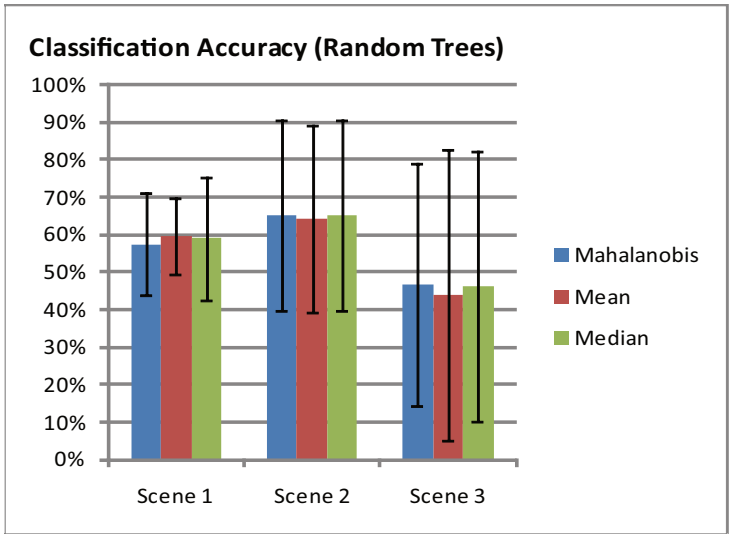


(b) Random Trees classifier

Figure 3.7: Accuracy of the head pose classification for SVM classifier and Random Trees classifier. **Generic** result is based on the generic dataset, **Proposed** result is based on our proposed method. The accuracy is based on normalized average of 8 classification classes. Standard deviations are indicated as error bars.



(a) SVM classifier



(b) Random Trees classifier

Figure 3.8: Accuracy of head pose estimation for each image selection method. **Mahalanobis** result is based on using Mahalanobis distance to select sample. **Mean** result is based on using mean image to select sample. **Median** result is based on using median image to select sample. The accuracy is based on normalized average of 8 classification classes. Standard deviations are indicated as error bars.

	1	2	3	4	5	6	7	8
1	0.15	0.25	0.03	0.13	0.23	0.2	0.03	0
2	0.03	0.1	0.05	0.3	0.23	0.2	0.1	0
3	0.03	0.15	0.03	0.13	0.38	0.25	0.05	0
4	0	0.03	0	0.08	0.53	0.38	0	0
5	0	0	0	0	0.53	0.48	0	0
6	0	0	0	0	0.13	0.83	0.05	0
7	0	0.03	0.08	0.03	0.13	0.6	0.15	0
8	0.1	0.03	0.03	0.08	0.2	0.2	0.3	0.08

(a) **SVM - Generic**

	1	2	3	4	5	6	7	8
1	0.25	0.5	0	0	0.1	0.13	0.03	0
2	0.1	0.28	0	0.2	0.33	0.1	0	0
3	0.03	0.2	0.03	0.33	0.4	0.03	0	0
4	0	0.05	0	0.05	0.9	0	0	0
5	0	0	0	0	0.85	0.15	0	0
6	0	0	0	0	0.35	0.58	0.08	0
7	0.13	0	0	0	0.1	0.43	0.35	0
8	0.25	0	0	0	0.05	0.15	0.55	0

(b) **Random Trees - Generic**

	1	2	3	4	5	6	7	8
1	0	0.8	0	0	0	0.13	0.08	0
2	0	0.95	0	0	0	0.05	0	0
3	0	0.6	0.18	0	0	0.23	0	0
4	0	0.08	0.05	0	0	0.88	0	0
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0.93	0.08	0
7	0	0.1	0	0	0	0.43	0.48	0
8	0	0.13	0	0	0	0.15	0.73	0

(c) **SVM - Imbalanced**

	1	2	3	4	5	6	7	8
1	0	0.95	0	0	0	0	0.05	0
2	0	1	0	0	0	0	0	0
3	0	0.8	0.1	0	0	0.1	0	0
4	0	0.08	0.1	0	0	0.8	0.03	0
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0.98	0.03	0
7	0	0.13	0	0	0	0.33	0.55	0
8	0	0.3	0	0	0	0.08	0.63	0

(d) **Random Trees - Imbalanced**

	1	2	3	4	5	6	7	8
1	0.58	0.08	0	0	0	0.03	0.03	0.3
2	0.23	0.65	0	0.03	0	0.03	0.03	0.05
3	0.03	0.23	0.43	0.18	0.03	0	0	0.13
4	0	0	0.05	0.7	0.13	0.08	0	0.05
5	0	0.05	0.03	0.2	0.38	0.3	0.05	0
6	0.03	0	0	0.08	0.05	0.5	0.33	0.03
7	0	0	0.03	0.03	0.03	0.15	0.65	0.13
8	0.08	0.05	0	0	0	0.03	0.4	0.45

(e) **SVM - Proposed**

	1	2	3	4	5	6	7	8
1	0.55	0.05	0.03	0	0	0	0	0.38
2	0.25	0.68	0	0.03	0	0	0	0.05
3	0.08	0.23	0.58	0.1	0.03	0	0	0
4	0	0	0.1	0.63	0.2	0.03	0	0.05
5	0	0.03	0.03	0.13	0.68	0.15	0	0
6	0	0.03	0	0.03	0.3	0.3	0.35	0
7	0.05	0.05	0	0	0.05	0.08	0.45	0.33
8	0.1	0.05	0	0	0	0	0.1	0.75

(f) **Random Trees - Proposed**

Figure 3.9: Confusion matrix of SVM classifier and Random Trees classifier using data from scene 1 with Mahalanobis distance as the image selection method. Each class number is defined in same way as in Figure 3.1. **Generic** result is based on using generic dataset as training data. **Imbalanced** result is based on scene-specific dataset without oversampling method. **Proposed** result is based on using the mean image as selection method.

Chapter 4

Transfer Learning with Biased Data Correction

Due to the fact that pedestrians do not walk uniformly over all directions, sometimes obtained dataset will be biased and head pose data in some classes might not be sufficient.

In this chapter, we introduce the idea to integrate a generic dataset, *i.e.*, manually labeled head images taken from a different scene, with the automatically generated scene-specific dataset. Higher accuracy cannot be expected for generic datasets as discussed above, however, the scene-specific dataset often has a highly biased distribution of training samples and does not necessarily have enough training samples for all directions.

4.1 Proposed Method

To complement the drawbacks of both datasets, our method seamlessly integrates the scene-specific dataset and a generic dataset via transfer learning. We tested the above idea in both classification and regression cases, and our method estimates head pose more accurately than using just a generic dataset, without the need to manually collect labeled data from the test scenes.

Transfer learning are techniques which address the problem of training estimators with low amount of train data for the task, called *target data*, by using a large amount of generic data, called *source data*. Source data and target data are different set of data captured with different settings but they are related in some way. In transfer learning problem, there are a lot of source data but target data are scarce, therefore, we try to make use of source data in order to solve the problem of scarcity of target data.

In our case, source data are head poses acquired from different scenes with different camera settings. The tasks and the general properties of head poses are, however, similar to our target data. Therefore, with transfer learning techniques, we could incorporate these generic dataset with our target data in order to improve the accuracy in classes which lack training data. Figure 1.2 shows a framework of our method.

In this chapter, for clarification purpose, we denote generic dataset as D_g and scene-specific dataset which we acquire from previous method in chapter 3 as D_s.

As mentioned earlier, walking directions contained in D_s are unevenly distributed. The accuracy of head pose estimation tends to be low for directions with few training samples. In chapter 3, we use oversampling technique to handle biased data distribution. However, with our proposed method, handling biased data can be incorporated with transfer learning seamlessly.

To improve the accuracy, generic datasets taken from different scenes are incorporated in our method. As we discussed, generic datasets cannot achieve accurate estimation unless the scenes are significantly similar. However, it is much easier to collect a uniformly distributed dataset in such cases. To complement the biased data in the area in which the data is scarce, a method based on transfer learning is used. A modified version of TrBagg Algorithm [19] which combines bagging with transfer learning is introduced to incorporate data from the generic dataset. TrBagg was originally designed for classification tasks. Here we introduce a modification on TrBagg so that it can be applied regression tasks, too.

The overall process is summarized in Algorithm 1. Given a generic dataset $D_g = \{(\boldsymbol{h}_k, p_k)\}$ with ground-truth head poses and the scene-specific dataset D_s, it outputs a set of head pose estimators F^*.

Let D be a merged dataset which consists of all of the samples in D_g and D_s together. First, training bags $B = \{B_1, \ldots, B_T\}$ are constructed by randomly selecting training samples from D. To select each training sample, a random number θ between 0 and 2π is generated and a training sample in D with the closest head pose to θ is added to the t-th bag B_t. The algorithm is repeated until the data for every bag is generated. A set of estimators $F = \{f_1, \ldots, f_T\}$ is then constructed from B, *i.e.*, each estimator f_i is trained with data from bag B_i. Another classifier, f_0, is trained with samples from the scene-specific D_s and added to F. This is done to prevent negative transfer when no useful data is found in D_g.

The empirical error of each estimator f'_t on D_s are then evaluated. In the case of classification task, it is defined as the number of incorrectly classified samples. In the case of regression, an average angular error is used. The classifiers are sorted by their respective errors in ascending order $\{f'_0, \ldots, f'_T\}$,

Algorithm 1 Transfer learning with biased data

Input: Generic dataset D_g, scene-specific dataset D_s, number of bags T, and bag size S

Merge datasets $D \leftarrow D_g \cup D_s$
for $t = 1, \ldots, T$ **do**
  $B_t \leftarrow \{\}$
  for $s = 1, \ldots, S$ **do**
    $\theta \leftarrow$ random number from 0 to 2π
    $P \leftarrow$ sample in D whose pose is closest to θ
    $B_t \leftarrow B_t \cup P$
  end for
end for
A set of estimators $F = \{f_1, \ldots, f_T\}$ is constructed from B by training f_t with data from bag B_t
Learn an estimator f_0 from the scene-specific dataset D_s; $F \leftarrow F \cup \{f_0\}$
Compute errors on D_s of each estimators in F and sort estimators in ascending order of these errors: $\{f'_0, \ldots, f'_T\}$
$\epsilon \leftarrow$ empirical error of f'_0 on D_s
$F' \leftarrow \{f'_0\}$; $F^* \leftarrow \{f'_0\}$
for $t = 1, \ldots, T$ **do**
  $F' \leftarrow F' \cup \{f'_t\}$
  $\epsilon' \leftarrow$ Average error of output from estimators in F' on D_s,
  if $\epsilon' \leq \epsilon$ **then**
    $F^* \leftarrow F'$
    $\epsilon \leftarrow \epsilon'$
  end if
end for

Output: F^*

wheref'_0 is an estimator with lowest error on D_s and f'_T the highest. This process is summarized in Figure 4.1

The empirical error of f'_0 is then chosen as the base error ϵ. The candidate set F' and the best result F^* are both set to $\{f'_0\}$. The bag with the next lowest empirical error is then selected and added to the candidates F', and the error ϵ' using F' is calculated. In the case of classification, the set F' estimates a value based on the majority vote of the classifiers in the set. Specifically, the estimated pose class θ^* is given by

$$\theta^* = \arg\max_{\theta \in \Theta} \sum_{f_t \in F'} I_t, \tag{4.1}$$

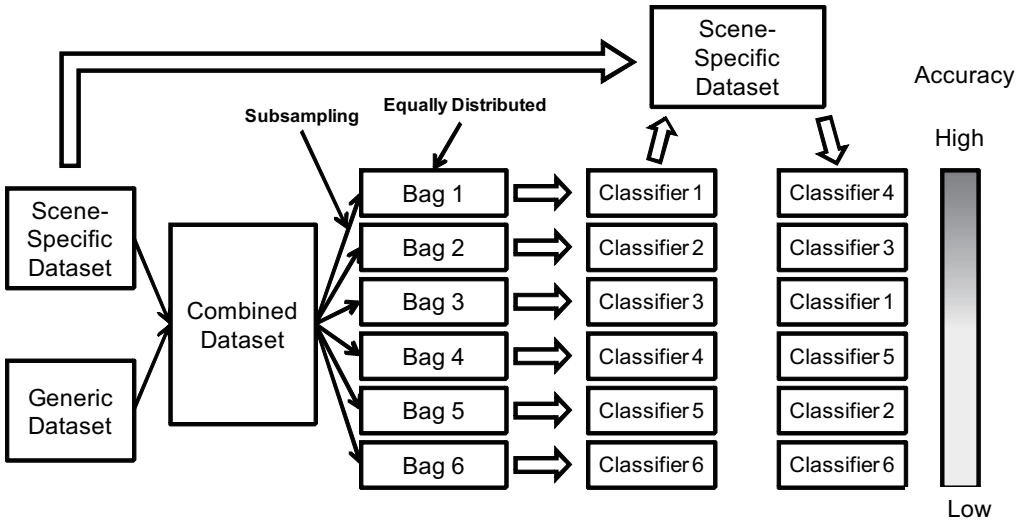


Figure 4.1: Overview of first part of the modified TrBagg method. Generic dataset and scene-specific dataset are combined and sampled into bags.An estimator is then created for each bag and evaluated with scene-specific dataset to measure the accuracy. The bags are then sorted by their accuracies.

where

$$I_t = \begin{cases} 1 & (f_t(\boldsymbol{h}^*) = \theta) \\ 0 & (\text{otherwise}) \end{cases}, \tag{4.2}$$

and $\boldsymbol{h}^*$ is an input head image. If vote count are the same for more than one class, the priority is given to votes of classifier with lower error value.

In the case of regression, the estimated head pose angle is the average of the values from each classifier in the set. Because the angle values are discontinuous at 2π degree, we compute θ^* such that

$$\sin(\theta^*) = \frac{1}{|F'|} \sum_{f_t \in F'} \sin(f_t(\boldsymbol{h}^*)), \tag{4.3}$$

and

$$\cos(\theta^*) = \frac{1}{|F'|} \sum_{f_t \in F'} \cos(f_t(\boldsymbol{h}^*)), \tag{4.4}$$

where $|F'|$ indicates a number of regressors in F'.

If ϵ' is smaller than ϵ, it means the combination of bags in F' is better than any combinations in the past and hence F' is stored as the best result $F*$ and the lowest empirical error ϵ is updated to ϵ'.

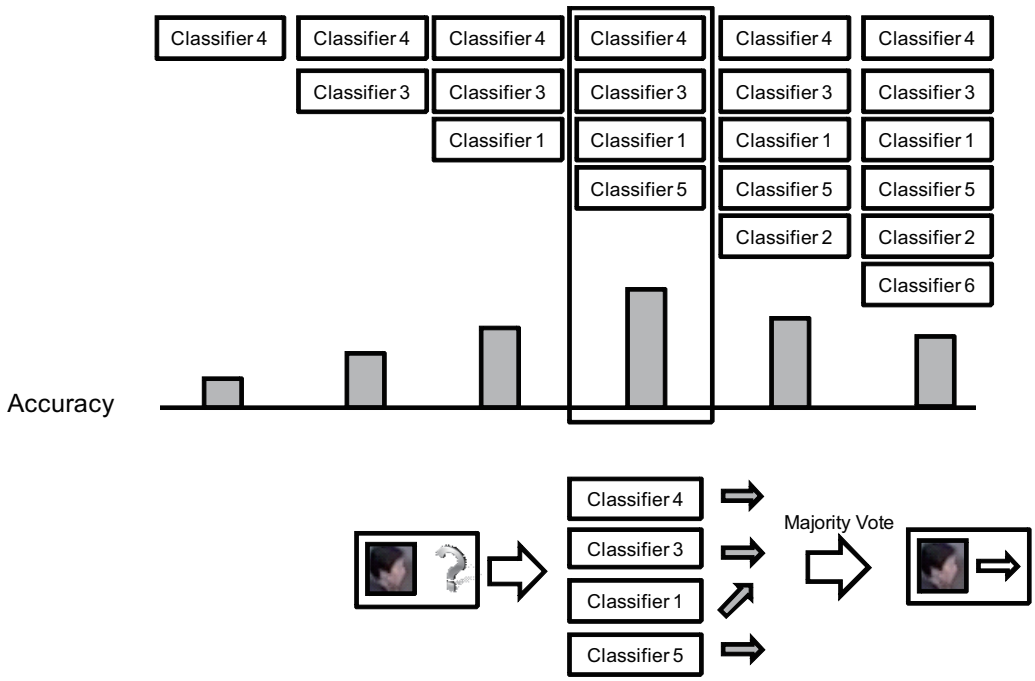


Figure 4.2: Overview of second part of the modified TrBagg method. Accuracies for sets of bags are evaluated and the set of bags with highest accuracy is then selected.

When every bag is considered, the set of classifiers with the lowest empirical error, F^* is output. For a given head image $\boldsymbol{h}^*$ the corresponding head pose p^* is estimated using Eq. (4.1) in the classification case, and Eq. (4.3) and Eq. (4.4) in case of regression. The process is summarized in Figure 4.2

4.2 Experimental Results

In this section, we present experimental results to demonstrate the effectiveness of our method. We conducted experiments using results obtained in the same manner as Chapter 3 as scene-specific dataset.

To construct a generic dataset, 1328 samples were taken from CAVIAR dataset [1]. The set is divided by head pose into 8 classes and each class contains $100 \sim 200$ images. Figure 4.3 shows examples of head images included in the generic dataset.

Figure 4.3: Example images in the CAVIAR dataset.

4.2.1 Classification Test

We first conducted a classification test using a linear SVM classifier [15]. To evaluate effectiveness of the proposed method, we compared three results: **Generic** result based on the generic dataset, **Naïve** result based on all of the scene-specific data, and **Proposed** result based on our method with outlier rejection and transfer learning.

Classification accuracy is summarized in Figure 4.2.1 with two types of measurement. The first type of measurement is a sample average which is computed as a simple average among test samples. Because of the camera position in our experimental setting, the number of test samples for each class is not equal. Hence another type of measurement, a class average, is also shown. In this case, sample averages are computed in each class first, and an average of these 8 averages is computed as the class average. Standard deviations are indicated as error bars.

As can be seen, the accuracy of classification using the generic dataset is significantly low. In contrast, even **Naïve** result achieved higher accuracy than the **Generic** result. However, the improvement of the sample average is much higher than the one of the class average. This is because the test data is taken from the same scene and has similar pose bias as in the scene-specific dataset. In the **Proposed** result, the class-average accuracy was also improved through transfer learning. Standard deviation also become small in the **Proposed** result.

Figure 4.2.1 more clearly shows the effect of the transfer learning. Confusion matrix for **Naïve** and **Proposed** result in the sequence 1 is shown in Figure 4.2.1. Each class number is defined in same way as in Figure 3.1. In Figure 4.5(a), it can be seen that classification accuracy is extremely low in class 1 and 5. They are corresponding to downward and upward directions which appear only infrequently in the video. By doing the transfer

learning, classification accuracy on these classes are improved as shown in Figure 4.5(b).

4.2.2 Regression Test

Next, we show results of regression using neural network [2]. Estimation accuracy is summarized in Figure 4.2.2. It shows average and standard deviation of estimation errors of three regressors: **Generic**, **Naïve** and **Proposed**, which are same as in Section 4.2.1.

Since ground-truth labels of the generic dataset is quite sparse, its performance on regression is lower and hence the contribution of the transfer learning cannot be expected so much. However, it can be seen clearly that the estimation accuracy was significantly improved in the **Naïve** result and the standard deviation was reduced through transfer learning in the **Proposed** result. Figure 4.7 shows some examples of the **Proposed** result. White lines indicate estimated head poses and red lines indicate manually-labeled ground truth poses.

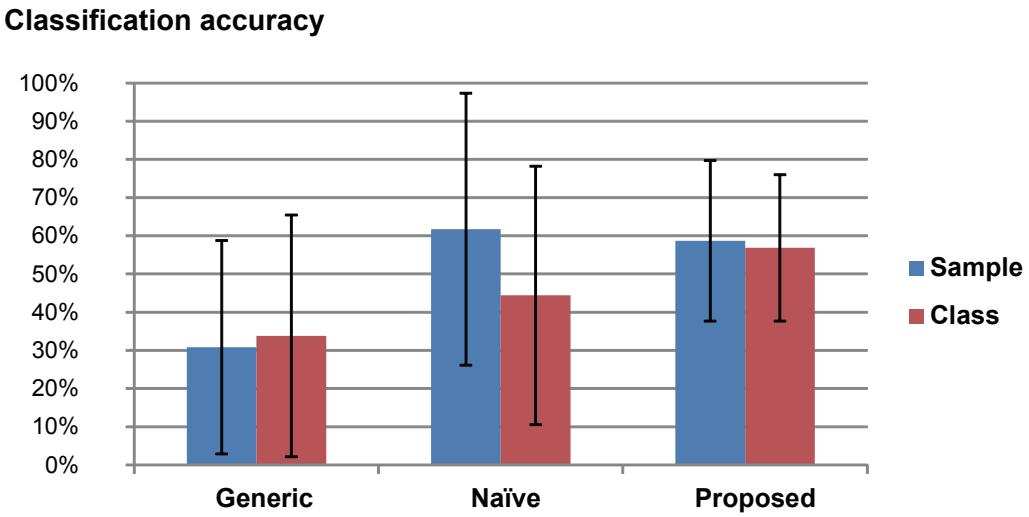
4.3 Conclusions

Because scene-specific dataset often has a highly biased distribution of training samples and does not necessarily have enough training samples for all directions, this chapter introduces the method which incorporates generic datasets into head pose estimation model.

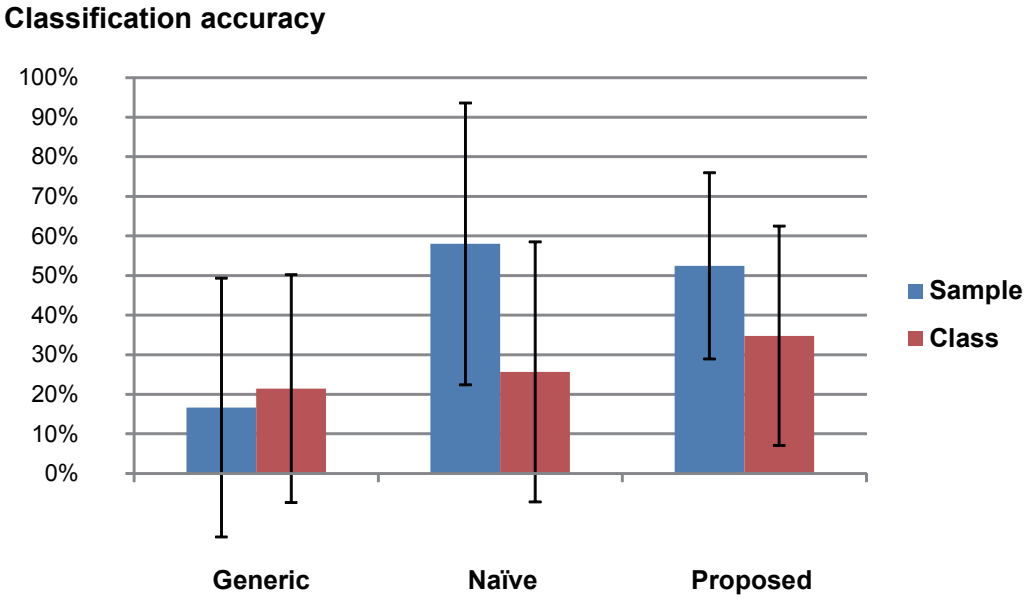
Although using only generic dataset itself do not yield efficient results, with the assumption that only some samples in the generic dataset are similar to target data, the algorithm is applied to select best sets of data which yield best accuracy based on available scene specific data.

This method also incorporates bagging, which is the technique to improve accuracy of estimation tasks. Bagging method reduces biases each estimator has by averaging the results from multiple estimators, thus make biases cancel out each other.

With this method, estimation accuracy improves for both classification and regressions tasks.



(a) Sequence 1



(b) Sequence 2

Figure 4.4: Accuracy of the head pose classification. **Generic** result is based on the generic dataset, **Naïve** result is based on all of the scene-specific data, and **Proposed** result is based on our method with outlier rejection and transfer learning. The sample average indicates average classification accuracy among test samples, and class average indicates the normalized average of 8 classification classes. Standard deviations are indicated as error bars.

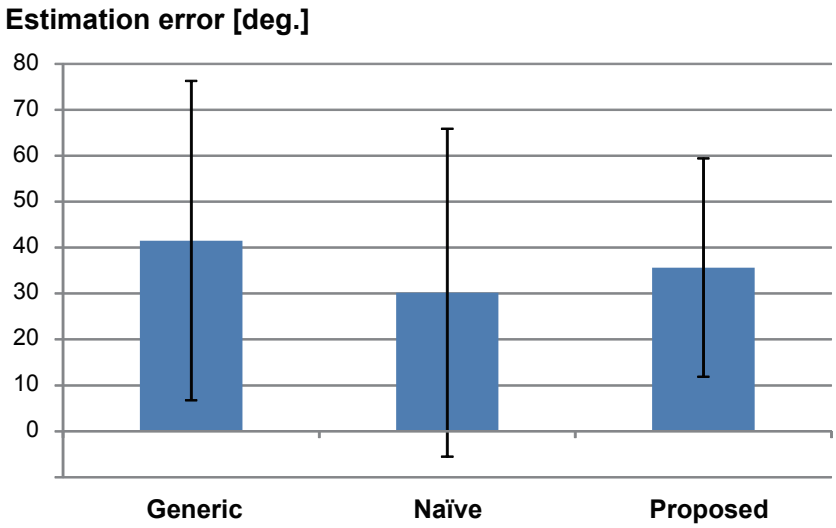
	1	2	3	4	5	6	7	8
1	0.00	0.00	0.25	0.00	0.00	0.13	0.50	0.13
2	0.00	0.23	0.18	0.00	0.05	0.00	0.05	0.50
3	0.00	0.05	0.85	0.04	0.04	0.01	0.00	0.00
4	0.00	0.00	0.16	0.37	0.11	0.37	0.00	0.00
5	0.00	0.00	0.13	0.31	0.06	0.50	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.90	0.10	0.00
7	0.00	0.00	0.11	0.00	0.00	0.02	0.81	0.06
8	0.00	0.17	0.11	0.00	0.06	0.00	0.33	0.33

(a) **Naïve**

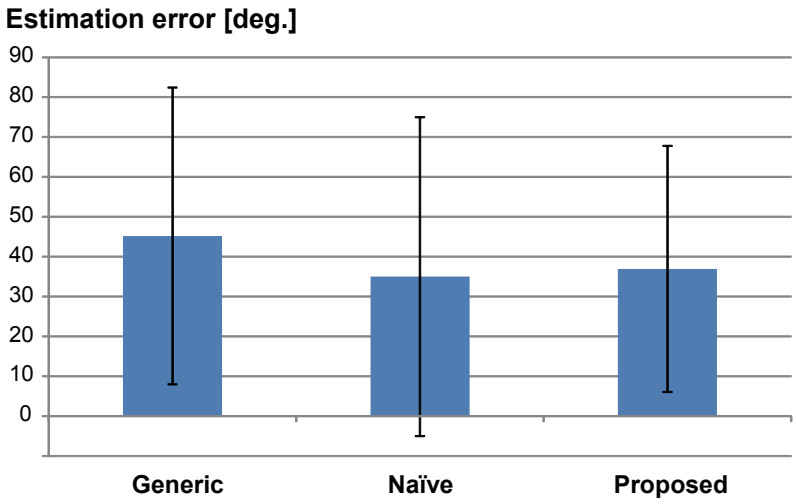
	1	2	3	4	5	6	7	8
1	0.50	0.38	0.13	0.00	0.00	0.00	0.00	0.00
2	0.23	0.68	0.05	0.00	0.05	0.00	0.00	0.00
3	0.00	0.13	0.57	0.23	0.03	0.04	0.00	0.00
4	0.00	0.00	0.00	0.47	0.05	0.42	0.05	0.00
5	0.00	0.00	0.06	0.25	0.19	0.50	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.90	0.10	0.00
7	0.00	0.00	0.02	0.00	0.03	0.10	0.68	0.18
8	0.22	0.11	0.11	0.00	0.00	0.00	0.00	0.56

(b) **Proposed**

Figure 4.5: Confusion matrix of the classification task. Each class number is defined in same way as in Figure 3.1.



(a) Sequence 1



(b) Sequence 2

Figure 4.6: Average error and standard deviation of the regression task. Definitions of **Generic**, **Naïve**, and **Proposed** are same as in Figure 4.2.1. Standard deviations are indicated as error bars.

Figure 4.7: Examples of head pose regression. White lines indicate estimated head poses and red lines indicate manually-labeled ground truth poses.

Chapter 5

Adaptive Scene Segmentation for Localized Head Pose Estimation

Even within the same scene, appearances of people can be drastically different due to lighting conditions or camera angles. Figure 5.1 shows difference in appearances of people in a scene for down-right class. Head pose images captured from nearby location tend to be similar while head poses taken at different locations tend to be different even if they are from the same class. Using one head pose estimator for each scene with such difference yields poor results.

5.1 Proposed Method

Instead of constructing a head pose estimator for each scene, we proposed to construct an estimator for each area which head poses are similar and the result show that the accuracy improves drastically compared to using only one estimator. Figure 1.3 shows a framework of our method.

Due to the observation that a scene can be divided into multiple areas in which head pose images are similar, we proposed a method to analyze those areas based on head pose samples taken from them.

A scene S can be viewed as a collection of non-intersecting areas $A_1, A_2, ..., A_n$ *i.e.*

$$S = \bigcup_{i=1}^{N} A_i, A_i \cap A_j = \emptyset, \forall A_i, A_j \in S, \tag{5.1}$$

In those areas, head pose samples are assumed to be similar.

Figure 5.1: Example of differences in appearances of people in a scene. Head pose images captured from nearby location tend to be similar while head poses taken at different locations tend to be different.

In order to segment a scene into multiple areas, we divide a scene into a completely connected graph $G = (V, E)$ with set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and edges E connect every pairs of nodes, $E = \{(v_i, v_j)|v_i, v_j \in N\}$. These nodes represent smallest unit to consider. Our area division will be more flexible with smaller nodes with the smallest being treating each pixel as a node, but doing so will yield large number of nodes and therefore increase the computational time of our method. Figure 5.2 shows example of dividing a scene into 10x10 nodes. Areas occupied by each node is called node area and is denoted by square around node center.

Our proposed method segments a scene into multiple areas by applying graph segmentation method to divide this graph into multiple areas which contain similar head samples.

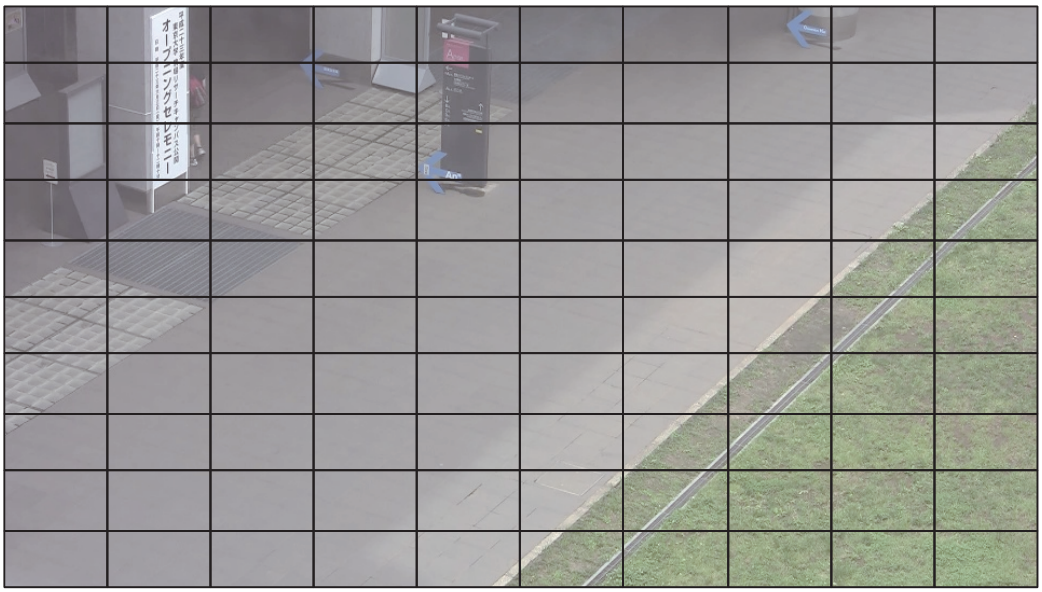


Figure 5.2: Example of dividing a scene into 10x10 nodes. Each node occupy node area defined by square around node center.

5.1.1 Graph Construction

In this section, we aim to define similarity for each node, in order to perform graph segmentation to divide the scene into areas. The similarity in Graph Segmentation is in the form of weight function $w(v_i, v_j)$, which defines similarity between node v_i and v_j. Weight function will be high for similar pair of nodes and low for dissimilar pairs. Use of this function will be discussed in section 5.1.2

With the dataset D captured with method in Chapter 3, we define D_{v_i} as head pose samples captured at node v_i. Also for clarification purposes, we defined $D_{v_i,c}$ as head pose samples of pose class $c \in C$ captured at node v_i, with C the set of all pose classes.

Our data-driven approach compares samples within each node area to determine similarity between each node and calculate weights for each pair of nodes effectively. In this section, the method to construct the graph and its corresponding weight function will be discussed in the top-down manner.

Weight Function

We first explore the weight function $w(v_i, v_j)$ between two nodes v_i and v_j. This weight function measures how similar two nodes are to each other.

Similarity weight of two nodes $w(v_i, v_j)$ are measured by two quantities. The first being sample weight between two nodes, denoted by $w_s(v_i, v_j)$. Sample weight measures similarity between samples captured from within each node. If samples from node v_i are similar to samples from node v_j, it is intuitive to assume that the two nodes are similar. Sample weight is defined as

$$w_s(v_i, v_j) = e^{\frac{-D_{node}(v_i, v_j)}{\sigma_A}}, \tag{5.2}$$

where $D_{node}(v_i, v_j)$ indicates node difference measure and σ_A a constant. $D_{node}(v_i, v_j)$ measures difference between two nodes. Node difference is high if two nodes are different and low if two nodes are similar. Details of this function will be discussed in the next section.

Another quantity of the node weight is distance weight, denoted by $w_d(v_i, v_j)$. Distance weight measures how likely node v_i are similar to node v_j. From the intuition that samples captured from nearby location tends to be more similar to samples captured from far away, we define this weight based on the location of two nodes. Distance weight w_d is defined based on the distance between two nodes as follows

$$w_d(v_i, v_j) = e^{\frac{-\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2}{\sigma_X}}, \tag{5.3}$$

where X_i and X_j are the position of node v_i and v_j respectively and σ_X a constant.

With sample weight and distance weight defined, the weight function between two nodes is established as a product of sample and distance weights, $w(v_i, v_j) = w_s(v_i, v_j) \cdot w_d(v_i, v_j)$.

Node Difference

As mentioned in the previous section, node difference measures difference in appearances of each node. Head poses captured in node areas with low difference are expected to be more similar than ones with high difference. We take data-driven approach to measure this difference by comparing head pose samples captured from each node. If head pose samples in two node areas are similar, it is intuitive to assume that future head poses taken from those areas will also be similar. Samples comparison is restricted to comparing samples with the same pose class due to the fact that head poses from different classes are not similar even if they are captured with the same setting.

Because some node areas might contain only a few head pose samples or even not containing any samples at all, defining difference between each node by comparing samples within that node alone yield poor result. Therefore,

we proposed to incorporate samples from nearby nodes into the equation. Because samples from nearby nodes should have less importance in calculation depending on how far they are from current node, we introduce importance function $f(v_1, v_2) \rightarrow [0, 1]$ which will be high if node v_1 and v_2 is near and low if they are far apart.

We define importance function as

$$f(v_i, v_j) = e^{\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma_X}}, \tag{5.4}$$

where X_i and X_j are the position of node v_i and v_j respectively with σ_X a constant.

With the importance function f defined, we derive the difference function between two nodes, D_{node}, as follows,

$$D_{node}(v_i, v_j) = \frac{\sum\limits_{v_1 \in G_i, v_2 \in G_j} D_{pair}(v_1, v_2) \cdot f(v_1, v_i) \cdot f(v_2, v_j)}{\sum\limits_{v_1 \in G_i, v_2 \in G_j} f(v_1, v_i) \cdot f(v_2, v_j)} \tag{5.5}$$

where $D_{pair}(v_1, v_2)$ denotes pair difference function. Pair difference function measures the difference between samples in two nodes.

Pair Difference

Next, we construct the measure of difference between head pose samples in each pair of nodes. As previously stated, it is intuitive to measure class difference by comparing samples with the same pose class in both nodes. Figure 5.3 shows an example of comparing head pose samples with the same class. Colored circles denote a sample within node v_i and v_j. Circles with the same color denote samples of the same class.

With this intuition, we define distance D_{pair} between two nodes as follows

$$D_{pair}(v_i, v_j) = \frac{\sum\limits_{c \in C, \boldsymbol{h_i} \in D_{v_i,c}, \boldsymbol{h_j} \in D_{v_j,c}} d(\boldsymbol{h_i}, \boldsymbol{h_j})}{\sum\limits_{c \in C} |D_{v_i,c}||D_{v_j,c}|}, \tag{5.6}$$

where $d(\boldsymbol{h_i}, \boldsymbol{h_j})$ denotes sample difference between two head pose samples $\boldsymbol{h_i}$ and $\boldsymbol{h_j}$.

Sample Difference

We will start the discussion from how to calculate the difference between each pair of head pose samples. Given a feature vector $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$ be the

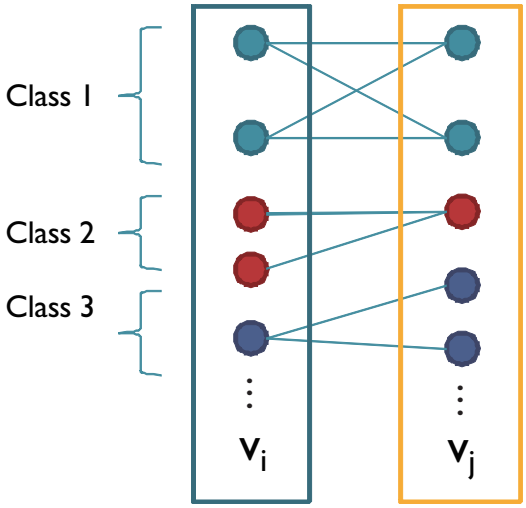


Figure 5.3: Example of comparing head pose samples with the same class. Colored circles denote a sample within node v_i and v_j. Circles with the same color denote samples of the same class.

feature vectors of two head pose samples, we define the difference between them as the weighted distance between each feature in the feature vectors

$$d(\boldsymbol{h_1}, \boldsymbol{h_2}) = \sqrt{(\boldsymbol{h_1} - \boldsymbol{h_2})^T M (\boldsymbol{h_1} - \boldsymbol{h_2})}, \tag{5.7}$$

where M is the diagonal matrix indicating importance of each feature. $M_{i,i}$ is the importance of ith feature in the vector $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$. $M_{i,i}$ is high if ith feature is important and has strong effect in distinguishing head poses. These are usually the feature which involve the area of face or human skin which has high importance in discriminating between head pose classes. Features with low importance might include background pixels which are irrelevant to head pose discrimination.

There are many approaches to obtain the matrix M. One of them is to train a Random Trees classifier [10] using dataset D and obtain this importance matrix.

With these functions defined, weight function $w(v_i, v_j)$ is constructed.

In the next step, we perform graph segmentation method to divide nodes V into N_R sets, $V_1, V_2, \ldots, V_{N_R}$, where N_R is a predefined constant.

5.1.2 Normalized Cut

Graph segmentation is the problem hat given a weighted undirected graph $G = (V, E)$ and weights $w(u, v)$ which defines similarity between node u and v, we seek to partition the graph into set of nodes $V_1, V_2, \ldots, V_N$ where similarity among nodes in a set V_i is high and across different sets V_i, V_j is low.

In graph theoretic language, partitioning a graph $G = (V, E)$ into two disjoint sets A and B is called graph cutting. Graph cut is defined as total weight of the edges that has been removed.

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v), \tag{5.8}$$

Minimum cut of a graph is the one that minimizes the *cut* value. Although this criteria is a good approximation for graph segmentation, minimum cut criteria favors cutting small set of isolated nodes in the graph.

Therefore, new measure which takes into account this problem called normalized cut [32] is created and defined as

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{5.9}$$

where $assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from nodes in A to all nodes in the graph.

To compute optimal partition for $Ncut(A, B)$, we can write the equation as follows

$$\min_{\boldsymbol{x}} Ncut(\boldsymbol{x}) = \min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T \boldsymbol{D} \boldsymbol{y}}, \tag{5.10}$$

where $\boldsymbol{x}$ be $N = |\boldsymbol{V}|$ dimensional vector where $x_i = 1$ if node i is in A and -1 otherwise, $\boldsymbol{x}\ \boldsymbol{d_i} = \sum_j w(i, j)$ and $\boldsymbol{D}$ be an NxN diagonal matrix with $\boldsymbol{d}$ on its diagonal.

If $\boldsymbol{y}$ is relaxed to take on real values, we can minimize (5.10) to generalized eigen value system,

$$(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y} = \lambda \boldsymbol{D} \boldsymbol{y}, \tag{5.11}$$

which can further be transformed into standard eigenvalue problem of

$$\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x} = \lambda \boldsymbol{x}. \tag{5.12}$$

Solving equation (5.12) requires solving a standard eigenvalue system, which takes $O(n^3)$ operations, where n is the number of nodes in the graph.

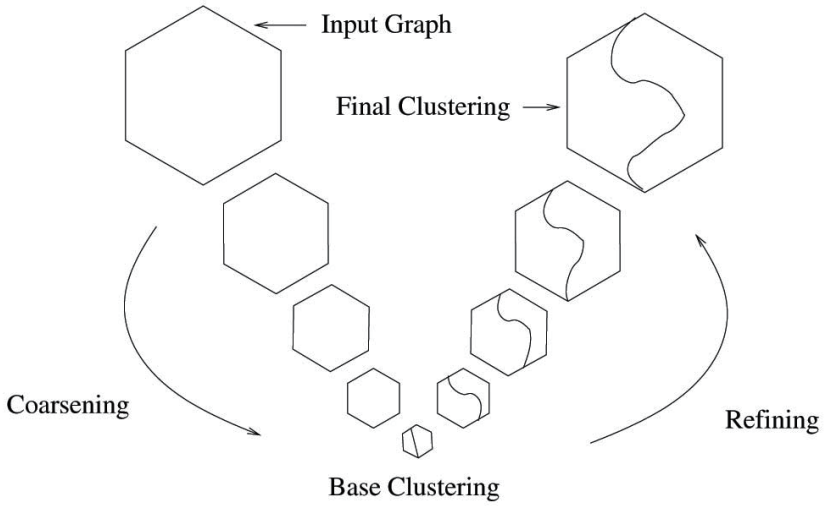


Figure 5.4: Illustration of the multilevel algorithm used in Dhilon *et al.* [13].

For the details on derivation of this equation and detailed discussions, we refer readers to [32].

5.1.3 Graclus

Because computing the optimal partition for normalized cut requires solving for eigen vectors thus requires computational time of $O(n^3)$, Dhilon *et al.* [13] developed a method called Graclus which does not require eigen vector computations with big data. We will briefly introduce the method. For more details, readers are referred to [13].

Graclus method's aim is to reduce computation speed while preserving the quality of graph cutting. This work proposes multilevel algorithm which divides graph clustering into 3 phases, coarsening, base clustering and refining phase. Figure 5.4 shows the illustration of the multilevel algorithm used in this method.

We denote an input graph $G_0 = (V_0, E_0, A_0)$, where V and E is vertices and edges of the graph and A as weight matrix such that A_{ij} equals to weight between node i and node j

Coarsening

Starting with the initial graph G_0, the coarsening phase repeatedly transforms the graph into smaller graphs $G_0, G_1, \ldots, G_m$ such that $|V_0| > |V_1| >$

$\cdots > |V_m|$. The transformation used in Graclus is called heavy edge coarsening.

The algorithm start with all nodes unmarked. Until all nodes are marked, randomly select an unmarked node x, select node y with highest edge weight from unmarked nodes adjacent to x, then mark x and y. If all of the neighbors of x have been marked, mark x and do not merge it with any node.

When all nodes are marked, the coarsening for current level is complete. Coarsening is done until number of nodes are smaller than a threshold. In this work, coarsening is done until the number of nodes are less than $5k$, where k is the number of desired clusters.

Base Clustering

In base clustering phase, conventional graph segmentation method is performed. Because only a few nodes remain, this can be done effectively and quickly.

This work proposes to use region-growing algorithm of Metis *et al.* [20]. This algorithm selects random nodes and grows regions around the node. The algorithm select initial random nodes several times and choose the best clustering result.

Refining

Given a graph G_i, this algorithm form a graph G_{i-1} by combining nodes in G_i. Using G_{i-1} as initialization for the graph, the algorithm use weighted kernel k-mean clustering to refine the graph.

The algorithm ends when the initial graph G_0 is refined. Computational time of this algorithm is up to 2,000 times faster than conventional methods.

With this method, we could divide areas in a scene S into areas $A_1, A_2, \ldots, A_{N_R}$ where A_i is the area occupied by nodes in V_i.

5.1.4 Classification

In classification stage, a classifier is created for each area $A_i, i = 1, 2, \ldots N_R$. The classifier in area A_i is trained with the head pose samples in node V_i. Specifically head pose samples,

$$D_{V_i} = \bigcup_{v_j \in V_i} D_{v_j}, \tag{5.13}$$

are used to train model for area A_i

5.1.5 Improved Data Sampling

As with the data sampling method in Chapter 3, we select one head pose sample per each line segment. Because a segment can pass through many nodes at once, sampling only one head pose sample does not yield efficient result as not every node this line segment pass through contain the head pose sample.

Therefore, we proposed to sample multiple head images from a line segment. After outliers rejection, head pose samples are selected from a line segment for every interval. Specifically, we capture head poses in line segments every c_t frames. Pose direction is set to be the line direction in the same manner as in Chapter 3, which mean poses acquired from the same line segment are assigned the same pose direction.

Figure 5.5 compares previous and proposed method. Figure 5.5(a) shows the previous sample method. Only one sample is selected for the line segment, making head pose samples reside in only one node. Figure 5.5(b) shows the proposed method. Head poses are sampled every c_t frames, making samples contained in multiple graph nodes.

5.2 Experimental Results

The algorithm with data obtained from Chapter 3. We divide the areas into 15x15 nodes.

We divided the tests into 2 categories, graph clustering test and classification test. In graph clustering test, the graph clustering algorithm results for varying R_N are shown. Classification test shows the efficiency of our proposed method in improving the classification accuracy.

5.2.1 Graph Clustering

With the obtained data, the weight for each pair of nodes are computed and are used divide a scene into N_R areas. Figure 5.6 shows the result of dividing scene 1 into multiple areas with similar head samples.

It can be seen that with $N_R = 2$, dark areas are separated from illuminated areas and with $N_R = 3, 4$ areas near the camera are starting to be separated from areas far from the camera. Note that we only use sample comparison and did not assume any kind of separation in advance.

5.2.2 Classification Test

Figure 5.7 show the result of adaptive head pose classification using the sampling method in Chapter 3. We set the value of the constant $\sigma_X = 10.0$. A classifier is created for each area obtained by graph clustering method. Head poses in each region are estimated using the corresponding classifier created using head pose samples in its region. Classification accuracy at $N_R = 1$ serves as baseline, as it equals to normal classification task where we use all pose samples to train a classification model.

Classification accuracy of the proposed head pose sampling method is also shown. We set the value of $c_t = 30$ which means head pose samples are taken every 30 frames in a line segment. Figure 5.8 shows the classification accuracy of the classifier trained with samples obtained with proposed method. The classification accuracy improve significantly over previous method due to increased head pose samples in every node.

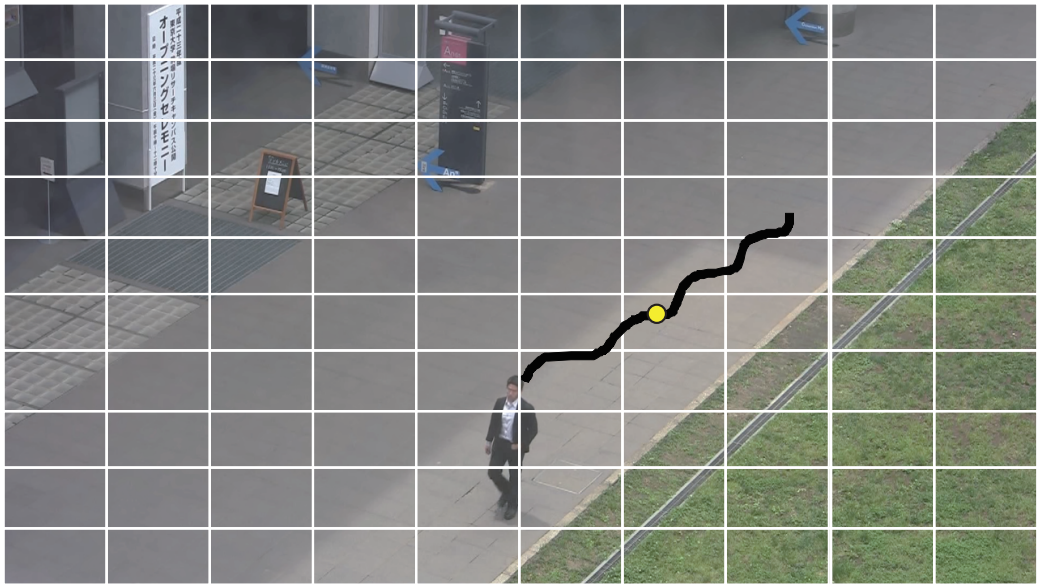
Compared to baseline, our proposed method significantly improves the classification accuracy.

5.3 Conclusions

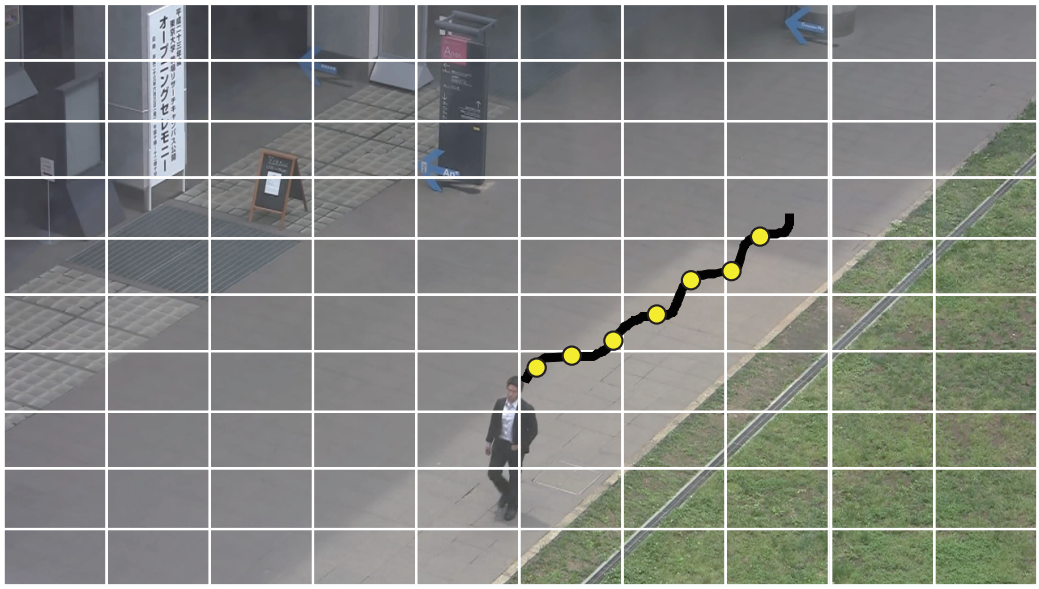
This method proposes head pose estimation method which also considers differences within scene. We divide a scene into regions and view them as graphs, with each small square region in a scene view as a graph node.

Weights between each node tells how similar each nodes are to each other and are defined based on similarity between head pose samples taken at the location of those nodes. Pairs of nodes with higher weights tend to be grouped together as the head pose samples within the node are more similar.

Our proposed method creates a classifier for every segmented region. Head poses will be classified by different classifiers based on the location those head poses are taken from. Experimental results show that our method which utilizes multiple classifiers achieve significantly better result than using only one classifier to classify the whole scene.



(a) Previous head pose sampling method



(b) Proposed head pose sampling method

Figure 5.5: Comparison of head pose sampling methods. White rectangles specify areas occupied by graph nodes. **Previous** head pose sampling method samples a head pose sample from each line segment. **Proposed** head pose sampling method. Head poses are sampled every c_t frames.

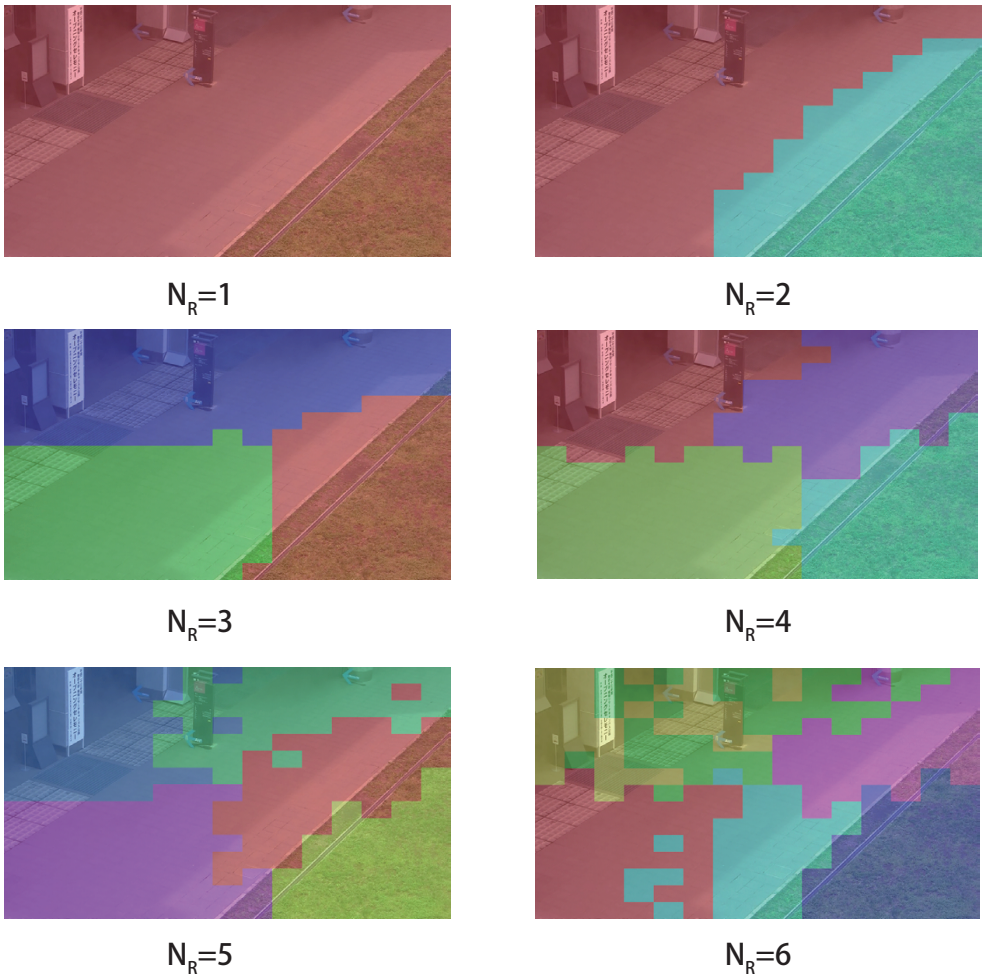


Figure 5.6: Result of dividing scene 1 into multiple areas with similar head samples with $N_R = 1$ to 6.

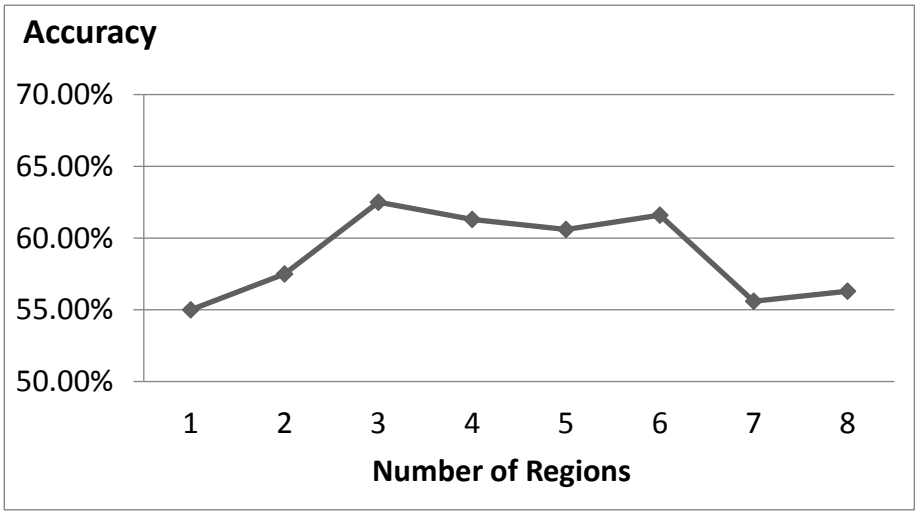


Figure 5.7: Result of head pose classification by creating a classifier for each region. Head poses in each region are estimated using the corresponding classifier created using head pose samples in its region. Accuracy at $N_R = 1$ serves as baseline.

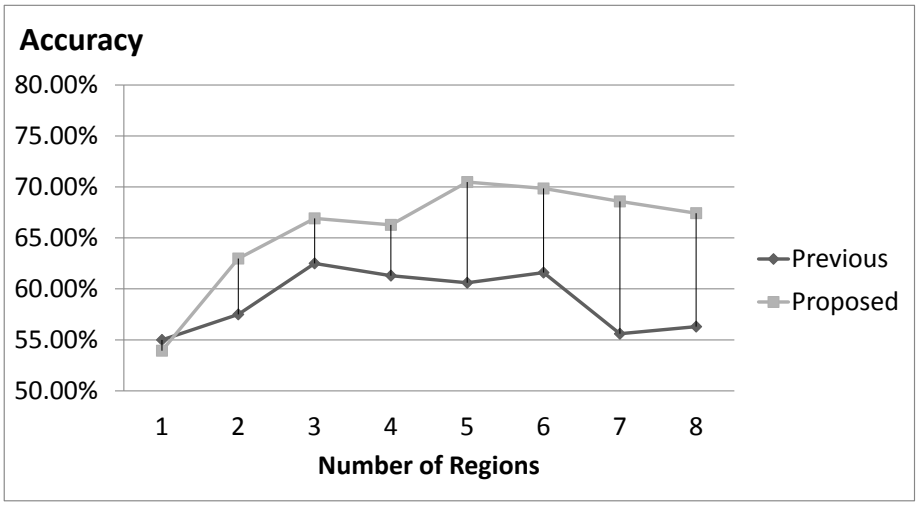


Figure 5.8: Result of head pose classification by using improved sampling method. **Previous** shows accuracy of the classifier trained with samples obtained from previous sampling method. **Proposed** shows accuracy of the classifier trained with samples obtained with proposed method. The accuracy significantly improves over previous method.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

Head pose estimation from low resolution image is crucial for many computer vision applications and is still being active in recent researches in computer vision. This research tackles one of the most important problems in head pose estimation from low resolution image.

Our work proposes head pose estimation method which aims at acquiring head pose samples automatically. This is done by tracking human head in target scene from prerecorded image sequences. We analyze tracked results to automatically acquire useful head pose samples based on the assumption that human turn to where they are walking most of the time. We also propose extensions to the method as follows.

In many cases the prerecorded image sequences are not long enough and cause the constructed head pose estimators to fail on some head pose classes. In this case, we proposed to integrate the generic dataset into the model and perform the modified TrBagg algorithm.

Also, even within the same scene, lighting conditions or camera angles might be different. We proposed the method to solve this problem by adaptively divide a scene into areas with similar head pose appearances. The division is based on the data contained within each area itself applied with graph segmentation method called Graclus.

Head pose estimation from low resolution images is itself a difficult task, and there is a much room for improvement in both feature description and classification/regression techniques. We believe these proposed methods would serve as a good basis for further developments of head pose estimation from low resolution images.

6.2 Future Works

We divide the aims of our future works into 2 categories, to improve head pose acquisition methods and to improve head pose estimation technique

Head Pose Acquisition

Head pose samples acquired with our proposed method still contains some erroneous head pose images even with outlier rejection methods. Erroneous samples are assumed to be of low amount in the model and are ignored.

One of the possible ways is to consider appearances of head pose samples taken with the proposed method again. If there are only a small number of erroneous samples, we could filter outliers out by removing head pose samples which appearances are not similar to other samples of the same class. It is also an interesting idea to integrate this with region division method in chapter 5 to further limit appearance difference to effectively filter out outliers.

Head Pose Estimation

Although head pose estimation from low resolution images are progressing rapidly, they are still a long way from perfect. Because good feature descriptor is one of the most important factors in head pose estimation and there is still no state-of-the-art feature which outperforms existing features, this topic is worth researching in as the future work.

Bibliography

[1] CAVIAR test case scenarios. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[2] Fast artificial neural network library. http://leenissen.dk/fann/wp/.

[3] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In Jean-Franois Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 39–50. Springer Berlin / Heidelberg, 2004.

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

[5] B. Benfold and I. D. Reid. Colour invariant head pose classification in low resolution video. In *Proc. British Machine Vision Conference (BMVC2008)*, 2008.

[6] B. Benfold and I. D. Reid. Guiding visual surveillance by tracking human attention. In *Proc. British Machine Vision Conference (BMVC2009)*, 2009.

[7] Ben Benfold and Ian Reid. Unsupervised learning of a scene-specific coarse gaze estimator. In *ICCV*, November 2011.

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. 10.1023/A:1010933404324.

[10] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[11] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, june 2005.

[13] I.S. Dhillon, Yuqiang Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1944 –1957, nov. 2007.

[14] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:315–354, 1973.

[15] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[16] N. Gourier, J. Maisonnasse, D. Hall, and 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)*, 2006.

[17] i LIDS Team. Imagery library for intelligent detection systems (i-LIDS); a standard for testing video based detection systems. In *Carnahan Conferences Security Technology, Proceedings 2006 40th Annual IEEE International*, pages 75 –80, October 2006.

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[19] T. Kamishima, M. Hamasaki, and S. Akaho. TrBagg: A simple transfer learning method and its application to personalization in collaborative tagging. In *Proc. 9th IEEE International Conference on Data Mining (ICDM2009)*, pages 219–228, 2009.

[20] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1998.

[21] B. D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[22] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004.

[23] L. P. Morency, A. Rahimi, N. Checka, and T. Darrell. Fast stereo-based head tracking for interactive environments. pages 390 –395, May 2002.

[24] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607 –626, 2009.

[25] J. Orozco, S. G. Gong, and T. Xiang. Head pose classification in crowded scenes. In *Proc. British Machine Vision Conference (BMVC2009)*, 2009.

[26] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345 –1359, oct. 2010.

[27] R. Pappu and P. A. Beardsley. A qualitative approach to classifying gaze direction. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 160–165, 1998.

[28] Victor Prisacariu and Ian Reid. fastHOG - a real-time GPU implementation of HOG. Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009.

[29] N. M. Robertson and I. D. Reid. Estimating gaze direction from low-resolution faces in video. In *Proc. 9th European Conference on Computer Vision (ECCV2006)*, volume 3952/2006, pages 402–415, 2006.

[30] N. M. Robertson, I. D. Reid, and J. M. Brady. What are you looking at? gaze estimation in medium-scale images. In *Proc. HAREM05, 16th British Machine Vision Conference, Oxford, September 2005*, 2005.

[31] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[32] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888 –905, aug 2000.

[33] R. Stiefelhagen. Estimating head pose with neural networks - results on the pointing04 icpr workshop evaluation data. In *Proceedings of the ICPR Workshop on Visual Observation of Deictic Gestures*, 2004.

[34] R. Stiefelhagen, J. Yang, and A. Waibel. Modeling focus of attention for meeting indexing based on multiple cues. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 13:928–938, 2002.

[35] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, CMU-CS-91-132, Carnegie Mellon University, 1991.

[36] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[37] Gary M. Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[38] Y. Wu and K. Toyama. Wide-range, person- and illumination-insensitive head orientation estimation. In *Proc. Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 183–188, 2000.

[39] Z. Zhang, Y. Hu, M. Liu, and T. Huang. Head pose estimation in seminar room using multi view face detectors. In *Proc. 1st International Evaluation Conference on Classification of Events, Activities and Relationships*, pages 299–304, 2007.

Publication List

1. Isarun Chamveha, Yusuke Sugano, Daisuke Sugimura, Teera Siriteerakul, Takahiro Okabe, Yoichi Sato, Akihiro Sugimoto , “ Appearance-Based Head Pose Estimation with Scene-Specific Adaptation”, in Proc. IEEE International Workshop on Visual Surveillance (VS2011), pp. 1713-1720, November 2011.

2. Isarun CHAMVEHA，菅野裕介，杉村大輔 , Teera SIRITEERAKUL, 岡部孝弘 , 佐藤洋一，杉本晃宏,“環境への自動適応を伴うアピアランスベース頭部姿勢推定”, 情報処理学会コンピュータビジョンとイメージメディア研究発表会, pp. 87-94, September 2011.