Domain Adaptation for Structured Regression*

Makoto Yamada[1], Yi Chang[1], Leonid Sigal[2]
[1]Yahoo Labs, 701 1st Ave., Sunnyvale, 94089, USA
[2]Disney Research, 4720 Forbes Ave., Pittsburgh, PA 15213, USA
`makotoy@yahoo-inc.com`, `lsigal@disneyresearch.com`

December 1, 2013

Abstract

Discriminative regression models have proved effective for many vision applications (here we focus on 3D full-body and head pose estimation from image and depth data). However, *dataset bias* is common and is able to significantly degrade the performance of a trained model on target test sets. As we show, covariate shift, a form of unsupervised domain adaptation (USDA), can be used to address certain biases in this setting, but is unable to deal with more severe structural biases in the data. We propose an effective and efficient semi-supervised domain adaptation (SSDA) approach for addressing such more severe biases in the data. Proposed SSDA is a generalization of USDA, that is able to effectively leverage labeled data in the target domain when available. Our method amounts to projecting input features into a higher dimensional space (by construction well suited for domain adaptation) and estimating weights for the training samples based on the ratio of test and train marginals in that space. The resulting augmented weighted samples can then be used to learn a model of choice, alleviating the problems of bias in the data; as an example, we introduce SSDA Twin Gaussian Process regression (SSDA-TGP) model. With this model we also address the issue of *data sharing*, where we are able to leverage samples from certain activities (e.g., walking, jogging) to improve predictive performance on very different activities (e.g., boxing). In addition, we analyze the relationship between domain similarity and effectiveness of proposed USDA vs. SSDA methods. Moreover, we propose a computationally efficient alternative to TGP (Bo and Sminchisescu, 2010), and it's variants, called the direct TGP (dTGP). We show that our model outperforms a number of baselines, on two public datasets: HumanEva and ETH Face Pose Range Image Dataset. We can also achieve 8 to 15 times speedup in computation time, over the traditional formulation of TGP, using the proposed direct formulation, with little to no loss in performance.

1 Introduction

Many problems in computer vision can be expressed in the form of discriminative (structured) predictions of real-valued multivariate output, $\boldsymbol{y} \in \mathbb{R}^{d_\mathrm{y}}$, from a high-dimensional multivariate input, $\boldsymbol{x} \in \mathbb{R}^{d_\mathrm{x}}$. A success of such methods in 3D full-body pose estimation is evident from recent results that use Microsoft Kinect sensor (Girshick *et al.*, 2011; Sun *et al.*, 2012); such discriminative methods have also proved effective for other problems, including image-based 3D pose (Bo and Sminchisescu, 2010; Kanaujia *et al.*, 2007; Shakhnarovich *et al.*, 2003; Sminchisescu *et al.*, 2006; Urtasun and Darrell, 2008), head pose (Fanelli *et al.*, 2011) and body shape (Chen *et al.*, 2011; Sigal *et al.*, 2007) estimation. The typical goal of discriminative regression methods is to learn a direct (and sometimes multi-modal) mapping, $\boldsymbol{f} : \mathbb{R}^{d_\mathrm{x}} \rightarrow \mathbb{R}^{d_\mathrm{y}}$, from features (e.g., computed from image or depth data) to pose (e.g., 3D position and orientation of the head, or full 3D pose of the body encoded by joint positions or joint angles). Despite success and large body of work, most approaches by-and-large still suffer from two important issues: *dataset bias* and *lack of data sharing* in learning – an inability to effectively leverage information from one, or

*To appear in IJCV Special Issue on Domain Adaptation for Vision Applications.

more, *source* motion(s) or pose types to improve performance on a different, *target*, pose type, which may have few training samples.

Dataset bias refers to the general observation that models trained on one dataset (or on a training partition of a dataset) may not perform well on another dataset (or disjoint test partition of the same dataset). Biases can come in variety of different forms and have been shown to be present even in very large and carefully collected datasets. Consider training a regression model that predicts position and orientation of a person's head from Kinect data as in Figure 1 (c). If the model is trained based on the data from subject A and then tested on subject B, the results may be sub-optimal because statistics of the facial features between the two subjects are likely to be different (e.g., subject B may have higher cheekbones which may lead the model to wrongly predict a higher pitch angle). Even if the model is trained on many subjects, the presence or absence of certain accessories (e.g., glasses) may cause similar performance degradation. Consider an even simpler case where the model is trained on many subjects, including a test subject under consideration. If the model cannot fit the training data perfectly, which is typical, the performance on the test subject may again be inferior because the learning algorithm would try to distribute errors among all subjects by minimizing the average error over the entire training set. Note, these issues are not specific to the problem of head pose estimation and exist in most regression problems.

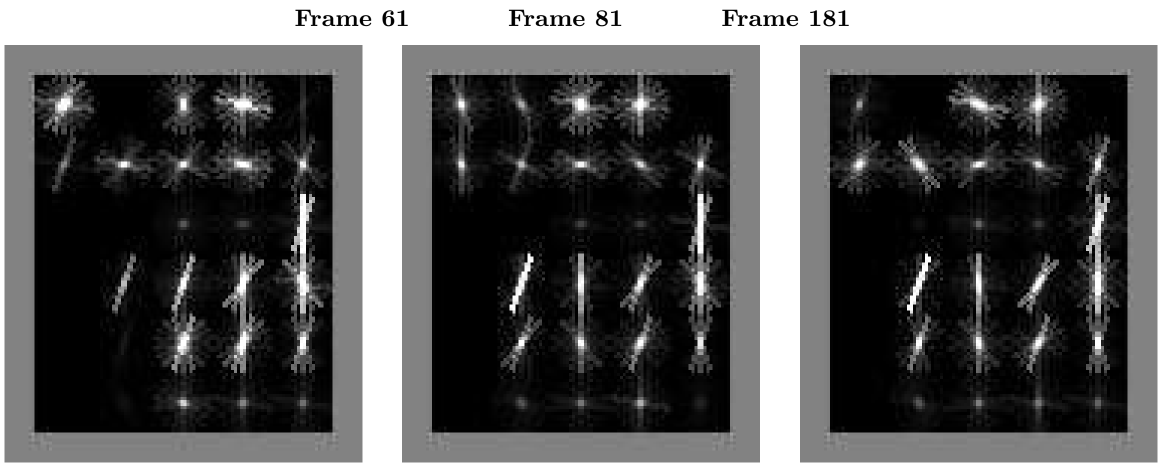
To address these issues we propose a new Semi-supervised Domain Adaptation (SSDA) approach, which is a generalization of the Unsupervised Domain Adaptation (USDA) we proposed earlier in Yamada *et al.* (2012). Our domain adaptation method allows us to easily adapt a model learned on a (source) training set of feature-pose pairs - $\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$, to a partially labeled (target) test set consisting of very few labeled examples - $\{(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}})\}_{j=1}^{n'_{\mathrm{te}}}$ (for which outputs are known) and many unlabeled samples for which outputs should be inferred - $\{(\boldsymbol{x}_j^{\mathrm{te}})\}_{j=n'_{\mathrm{te}}}^{n_{\mathrm{te}}}$. When labeled test samples (in the target domain) are unavailable, it effectively reduces to the original USDA formulation.

Our SSDA approach is surprisingly simple yet effective. First, both target and source samples are transformed into a common higher-dimensional space, through feature augmentation (Daumé, 2007); this augmentation in-itself *aligns* the two spaces[1]. Second, we apply importance weight estimation to correct for sampling bias in the augmented feature space. Finally, we use the resulting weighted augmented samples to learn a non-linear multivariate regression model in the form of Importance Weighted Twin Gaussian Processes (Yamada *et al.*, 2012). Moreover, we propose a computationally efficient alternative to TGP (Bo and Sminchisescu, 2010), that we call direct TGP (dTGP). The benefit of dTGP is that the learning and inference can be carried out by using only simple linear algebra. We apply the proposed approach to two problems (see Figure 1): (1) 3D pose estimation from images on the HumanEva dataset (Sigal and Black, 2006) and (2) 3D head pose estimation based on depth data using the ETH Face Pose Range Image Dataset (Breitenstein *et al.*, 2008).

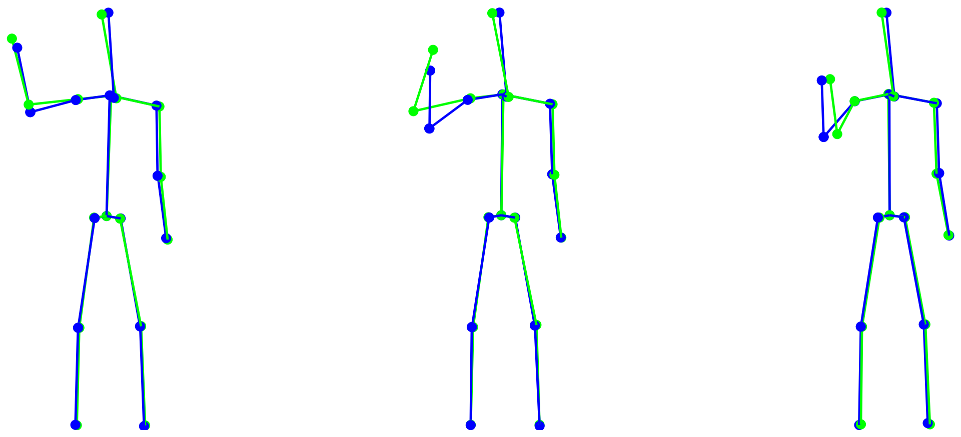
The issue of *dataset bias* was partially addressed in our preliminary work in Yamada *et al.* (2012). As we observe in Yamada *et al.* (2012), the key assumption that training and test samples come from the same underlying density (i.e., $(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}}) \sim p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$, $(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}}) \sim p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$), is often flawed, even for large datasets; this fact leads to biased models that perform sub-optimally on the test set. In Yamada *et al.* (2012), we have proposed a simple unsupervised approach for removing certain biases, most notably those that adhere to the assumption of covariate shift, where both the training and test sets are defined on the same domain (overlap) but the density of samples may be different. This setting is well suited for reducing effects of sampling bias. However, as we show here, the entirely unsupervised approach may be ineffective in dealing with more severe biases (e.g., those that induce structured changes between the training and test data) that make the training and test domains largely disjoint (e.g., see Figure 3 (a)). Here we expand on the findings in Yamada *et al.* (2012), and propose a much more aggressive semi-supervised approach that is able to work even in the cases where the earlier formulation becomes less effective. It's important to note that the proposed approach is a strict generalization. It can be reduced to our earlier unsupervised formulation in Yamada *et al.* (2012) by choosing appropriate model parameter.

[1]We use the term *alignment* loosely here, as in practice, the spaces are not explicitly *aligned*, but rather the augmented space is by construction better suited for learning the adapted model; this effect is achieved by implicitly assigning higher importance to labeled test samples over training samples.

(a)



(b)



(c)

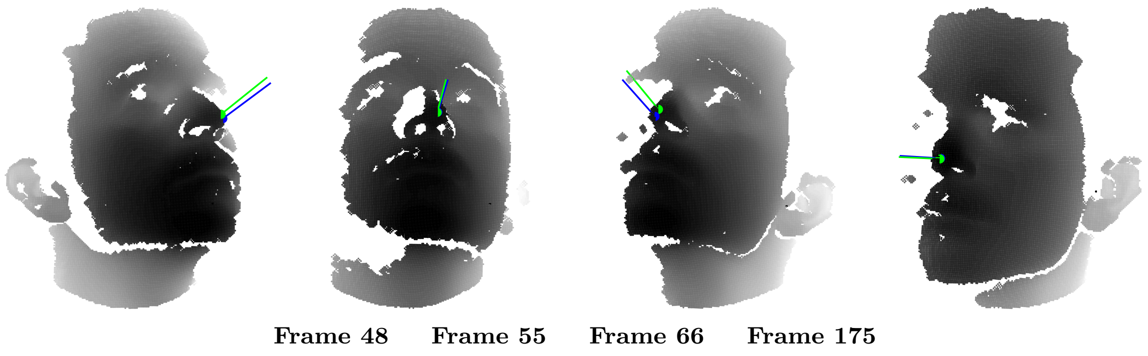


Figure 1: **Illustration of results:** HoG features for the *gesture* motion from HUMANEVA-I dataset (a) along with the corresponding poses inferred using proposed SSDA method (blue) and the ground truth (green) in (b); results of SSDA on the head pose estimation in ETH Face Pose Range Image Dataset is illustrated in (c).

The issue of *data sharing* has been around for some time in object detection and categorization community, but to our knowledge, has not been explored in the context of 3D pose estimation (i.e., structured prediction problems). In object detection, it is often argued that for certain classes of objects the data is scarce and information transfer from other categories (e.g., in the form of priors (Miller *et al.*, 2000), feature sharing (Torralba *et al.*, 2004) or data sample sharing (Lim *et al.*, 2011)) is necessary to regularize the performance. In pose estimation, one is typically interested in learning a single regression function. However, we argue that similar issues hold, in that for certain parts of the pose space the samples may be scarce (e.g., we may only have 2–3 annotated *boxing* postures in the training set, but thousands of *walking* and *jogging* postures). In this case utilizing samples from other parts of the space to regularize performance in a smart way, as we show, can also lead to improvements. Because our approach can deal with large biases in the data, we can simply treat data sharing as a form of a bias (e.g., where we let samples from walking and jogging constitute our training set and samples from boxing our test set). It is worth noting that in the object detection and categorization community, in contrast, the two problems of data sharing and bias are often addressed separately.

Another potential use case for our semi-supervised method is general model adaptation. Imagine a system (e.g., Microsoft Kinect) shipped with a pose estimation model trained on a large dataset of exemplars (e.g., (Shotton *et al.*, 2011) quotes 15 base body mesh shapes and a total of about 500,000 annotated samples). While effective, this model may not be optimally tuned for recognizing poses specific to some activity that one may want to employ in a game design (e.g., fishing) or to a specific user (e.g., 6 year old girl); by cueing and recording the user in a set of predefined postures (thereby getting a few annotated samples for the target test set) the performance of the general model can be improved.

1.1 Contributions and Core Findings

We expand on our original unsupervised domain adaptation (USDA) formulation for regression, in Yamada *et al.* (2012), by proposing a new semi-supervised domain adaptation (SSDA) approach. Proposed SSDA approach when used in absence of labeled target domain samples, and/or under certain parameter settings, reduces to our earlier formulation of USDA. Both, USDA and the more general SSDA are amenable to most discriminative and structured regression/prediction models and problems. The proposed SSDA method is simple, efficient and effective. It amounts to projecting input features to a higher dimensional space and appropriately weighting them to alleviate domain biases; the resulting transformed and weighted samples are then used for learning. Using USDA and SSDA as the basis, we propose and explore new forms of Twin Gaussian Processes regression (USDA-TGP) and (SSDA-TGP).

Further, we propose a measure of domain similarity, between the source and target domains, which allows us to explore the tradeoffs between effectiveness of USDA-TGP and SSDA-TGP. As a consequence, we are able to show that the proposed SSDA-TGP model is more effective in removing large structural biases in data and in promoting data sharing during learning, as is illustrated in two applications: 3D pose estimation from images and 3D head pose estimation from depth data. The unsupervised variant, USDA-TGP, on the other hand, is more effective in removing smaller biases such as selection bias. Moreover, we propose a structured prediction method we call direct TGP (dTGP), which is a computationally efficient approximation to TGP and it's variants. We show that dTGP is nearly equally effective, but is 8 to 15 times faster than standard TGP formulation (and corresponding variants).

2 Related Work

Our work touches on a number of topics in both computer vision and machine learning, including, discriminative (structured) regression, pose estimation, transfer learning, domain adaptation, and dataset bias.

Discriminative regression: Over the past 10 years many methods have been introduced that include both parametric (e.g., conditional mixture of experts (Kanaujia *et al.*, 2007; Sminchisescu *et al.*, 2006)) and non-parametric (e.g., nearest neighbor regression (Shakhnarovich *et al.*, 2003), linear locally-weighted regression (Shakhnarovich *et al.*, 2003), regression forests (Sun *et al.*, 2012), local Gaussian process regression (Urtasun and Darrell, 2008), twin Gaussian processes regression (Bo and Sminchisescu, 2010), etc.) models. A variety of feature representations and learning architectures have also been explored. We build on this literature by formulating a new SSDA variant of twin Gaussian process regression (Bo and Sminchisescu, 2010) with histogram of oriented gradients (HoG) as features.

Domain adaptation: The first step of our SSDA domain adaptation approach relates closely to the EasyAdapt method of (Daumé, 2007), where domain adaptation is achieved by projecting the source and target data into a higher dimensional space, through augmentation. This projection in-itself facilitates domain adaptation. However, we do not stop there, but also allow these samples to be weighted, to account for additional sampling biases in the data. For more detailed overview of domain adaptation we refer reader to (Jiang, 2007; Pan and Yang, 2010). The issue of dataset bias has recently emerged as a serious problem in object categorization, with (Torralba and Efros, 2011) showing that significant biases exist in all current datasets.

Domain adaptation in object categorization: Since in our mapping each feature in the original problem is mapped into three versions of it: a *general* version, a *source-specific* version and *target-specific* version, our approach also relates to Khosla *et al.* (2012) where dataset bias in object recognition is addressed by learning a generalized model and dataset specific models within the max-margin framework. Authors in Khosla *et al.* (2012) also point out the relationship of their method, and by induction of our method, to a problem of multi-task learning (Evgeniou and Pontil, 2004). Alternative methods for domain adaptation in object recognition space include semi-supervised metric learning (Saenko *et al.*, 2010; Kulis *et al.*, 2011) and un-supervised domain shift (Gopalan *et al.*, 2011). The key difference with these methods is that our approach is simpler (instead of learning a metric space as in Saenko *et al.* (2010); Kulis *et al.* (2011), we define our transformations in closed form), more general (in a sense that it can work with nearly any regression model), and works with multivariate (structured) real-valued outputs.

Transfer learning for sharing: Information transfer for enhancing performance of object detectors on categories where data is sparse dates to at least 2000. Many approaches exist that include different forms of priors (Miller *et al.*, 2000), feature sharing (Torralba *et al.*, 2004) or use intermediate representations (like attributes). Conceptually (but not mathematically) our approach is most similar to the idea of sample sharing introduced in Lim *et al.* (2011), where an object detector can opportunistically borrow transformed samples from other classes to enhance it's performance on a target class. Similar in spirit, our learning procedure is able to *borrow* transformed weighted samples from motions of other types (as is illustrated by our ability to perform motion transfer experiments in Figure 5 (c,f,g)).

3 Unsupervised Domain Adaptation (USDA) for Regression

In this section, we introduce our unsupervised domain adaptation (USDA) method for 3D pose estimation, after our original formulation in Yamada *et al.* (2012). This method amounts re-weighting of training instances based on the ratio of their probabilities under the test and training marginals. Note that while we focus on 3D body and head pose estimation in this paper, the proposed approach is applicable to any regression problem.

Let $\mathcal{X}^{\mathrm{tr}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of training image feature vector $\boldsymbol{x}^{\mathrm{tr}}$, $\mathcal{Y}^{\mathrm{tr}}(\subseteq \mathbb{R}^{d_{\mathrm{y}}})$ be the domain of training pose vector $\boldsymbol{y}^{\mathrm{tr}}$, and $\mathcal{X}^{\mathrm{te}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of testing image feature vector $\boldsymbol{x}^{\mathrm{te}}$. Suppose we are given n_{tr} i.i.d. training image-pose feature pairs and n_{te} i.i.d. test image feature vectors,

$$\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}}) \mid \boldsymbol{x}_i^{\mathrm{tr}} \in \mathcal{X}^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}} \in \mathcal{Y}^{\mathrm{tr}},\ i = 1, \ldots, n_{\mathrm{tr}}\},$$
$$\{\boldsymbol{x}_j^{\mathrm{te}} \mid \boldsymbol{x}_j^{\mathrm{te}} \in \mathcal{X}^{\mathrm{te}},\ j = 1, \ldots, n_{\mathrm{te}}\},$$

drawn from distributions with the densities $p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$ and $p_{\mathrm{te}}(\boldsymbol{x})$ respectively.

The final goal of USDA for 3D pose estimation is to learn a function $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})$ with low expected pose error, in the target domain, based on the training image-pose feature pairs and test image feature vectors.

Learning of model parameters, $\boldsymbol{\Theta}$, amounts to solving the following optimization problem:

$$\min_{\boldsymbol{\Theta}} \left[\iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})) p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \right]. \quad (1)$$

Note, we need to minimize the error over the test distribution p_{te} (not over the training distribution p_{tr}).

Because in the unsupervised setting we do not have test image-pose feature pairs, the optimization of Eq.(1) is not feasible. Thus, instead, we consider the following covariate shift adaptation problem (Shimodaira, 2000):

$$\min_{\boldsymbol{\Theta}} \left[\iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \right], \tag{2}$$

where $w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ is the *importance weight* function.

The advantage of this USDA formulation is that we can reduce sample selection bias, with the use of large number of test image features, through covariate shift adaptation.

Our proposed method consists of the following two steps that we describe in turn:

(i) Estimation of the importance weights $w(\boldsymbol{x})$ using RuLSIF (see Section 3.1).

(ii) Importance weighted learning of parameters $\boldsymbol{\Theta}$ of $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})$ from weighted augmented feature-pose pairs (see Section 5 and Appendix A).

3.1 Importance Weight Estimation

The importance weight may be estimated by independently estimating densities $p_{\mathrm{tr}}(\boldsymbol{x})$ and $p_{\mathrm{te}}(\boldsymbol{x})$ from training and test feature vectors and then taking their ratio. However, density estimation is known to be a hard problem and taking the ratio of estimated densities tends to increase the estimation error (Sugiyama *et al.*, 2008). Thus, this two-step approach is not appropriate in practice. We use a direct density-ratio estimation method that allows us to directly learn the importance weight function without going through density estimation. We exploit an importance weight estimation method called the *relative unconstrained least-squares importance fitting* (RuLSIF) (Yamada *et al.*, 2013).

Let us first define the *relative importance weight* (Yamada *et al.*, 2013):

$$w_\alpha(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{(1-\alpha)p_{\mathrm{te}}(\boldsymbol{x}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x})}, \ 0 \leq \alpha \leq 1, \tag{3}$$

where α is a tuning parameter to control the *adaptiveness* to the test distribution. Note that $p_{\mathrm{tr}}(\boldsymbol{x})$ here is a distribution over all labeled samples (training and test; if labeled test samples are available). If $\alpha = 0$ (i.e., $w_0(\boldsymbol{x}) = 1$) gives no adaptation, while $\alpha = 1$ (i.e., $w_1(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$) gives the full adaptation from $p_{\mathrm{tr}}(\boldsymbol{x})$ to $p_{\mathrm{te}}(\boldsymbol{x})$; $0 < \alpha < 1$ will give an intermediate estimator[2].

Let $\mathcal{X}^{\mathrm{tr}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of training image feature vector $\boldsymbol{x}^{\mathrm{tr}}$ and $\mathcal{X}^{\mathrm{te}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of test image feature vector $\boldsymbol{x}^{\mathrm{te}}$. Suppose we are given n_{tr} and n_{te} i.i.d. training and test image feature vectors, $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$, $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, drawn from distributions with densities $p_{\mathrm{tr}}(\boldsymbol{x})$ and $p_{\mathrm{te}}(\boldsymbol{x})$, respectively.

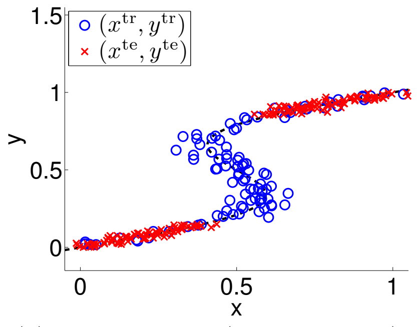
The final goal of relative importance weight estimation is to estimate the relative importance weight based on the training and test image features.

Let us model the relative importance weight $w_\alpha(\boldsymbol{x})$ by the following kernel model:

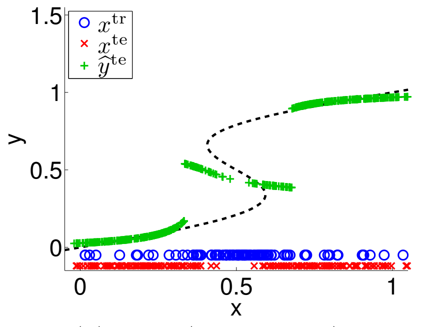
$$w_\alpha(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{\ell=1}^{n_{\mathrm{te}}} \theta_\ell \kappa(\boldsymbol{x}, \boldsymbol{x}_\ell^{\mathrm{te}}), \tag{4}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n_{\mathrm{te}}}$ are parameters to be learned from data samples, $^\top$ denotes the transpose, $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_\ell^{\mathrm{te}}\|^2}{2\tau^2}\right)$ is the Gaussian kernel and τ (> 0) is the kernel bandwidth.

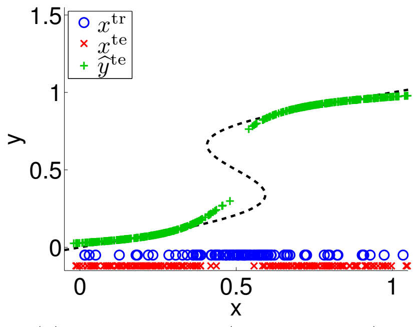
[2] $\alpha = 1$ (i.e., $w_1(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$) gives the full adaptation from $p_{\mathrm{tr}}(\boldsymbol{x})$ to $p_{\mathrm{te}}(\boldsymbol{x})$. However, since the importance weight $w_1(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ can diverge to infinity under a rather simple setting, the estimation of $w_1(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ is unstable and the covariate shift adaptation tends to be unstable (Shimodaira, 2000). To cope with this instability issue, setting α to $0 < \alpha < 1$ is practically useful for stabilizing the covariate shift adaptation, even though it cannot give an unbiased model under covariate shift (Yamada *et al.*, 2013).



(a) Data samples (ROD = 0.177)



(b) TGP (MSE=0.038)



(c) USDA-TGP (MSE=0.002)

Figure 2: Predicted outputs y using TGP (b), and USDA-TGP (c) in green. Samples from the model $x = y + 0.3\sin(2\pi y) + e$ where $e \sim \mathcal{N}(0, 0.05^2)$ are illustrated in (a); $\circ$ and $\times$ are training and test samples respectively (for clarity we also illustrate marginals $p_{\text{tr}}(\boldsymbol{x})$ and $p_{\text{te}}(\boldsymbol{x})$ in (b) and (c) bottom). Note that the input-output test samples are not used in the training of TGP and the output test samples are not used in the training USDA-TGP, they are plotted for illustration purposes only.

The parameters $\boldsymbol{\theta}$ in the model $w_\alpha(\boldsymbol{x};\boldsymbol{\theta})$ are determined so that the following expected squared-error J is minimized:

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \frac{1}{2}\mathbb{E}_{q_\alpha(\boldsymbol{x})}\left[(w_\alpha(\boldsymbol{x};\boldsymbol{\theta}) - w_\alpha(\boldsymbol{x}))^2\right], \\
&= \frac{(1-\alpha)}{2}\mathbb{E}_{p_{\text{te}}(\boldsymbol{x})}\left[w_\alpha(\boldsymbol{x};\boldsymbol{\theta})^2\right] + \frac{\alpha}{2}\mathbb{E}_{p_{\text{tr}}(\boldsymbol{x})}\left[w_\alpha(\boldsymbol{x};\boldsymbol{\theta})^2\right] \\
&\quad - \mathbb{E}_{p_{\text{te}}(\boldsymbol{x})}[w_\alpha(\boldsymbol{x};\boldsymbol{\theta})] + \text{Const.},
\end{aligned}$$

where $q_\alpha(\boldsymbol{x}) = (1-\alpha)p_{\text{te}}(\boldsymbol{x}) + \alpha p_{\text{tr}}(\boldsymbol{x})$, and we used $w_\alpha(\boldsymbol{x})q_\alpha(\boldsymbol{x}) = p_{\text{te}}(\boldsymbol{x})$ in the third term.

Approximating the expectations by empirical averages, we obtain the following optimization problem:

$$\widehat{\boldsymbol{\theta}} = \mathop{\text{argmin}}_{\boldsymbol{\theta}\in\mathbb{R}^{n_{\text{te}}}} \left[\frac{1}{2}\boldsymbol{\theta}^\top\widehat{\boldsymbol{H}}\boldsymbol{\theta} - \widehat{\boldsymbol{h}}^\top\boldsymbol{\theta} + \frac{\nu}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}\right], \tag{5}$$

where $\nu\boldsymbol{\theta}^\top\boldsymbol{\theta}/2$ is included to avoid overfitting, and ν (≥ 0) denotes the regularization parameter. $\widehat{\boldsymbol{H}}$ is the $n_{\text{te}} \times n_{\text{te}}$ matrix with the (ℓ, ℓ')-th element

$$\begin{aligned}
\widehat{H}_{\ell,\ell'} &= \frac{(1-\alpha)}{n_{\text{te}}}\sum_{i=1}^{n_{\text{te}}} \kappa(\boldsymbol{x}_i^{\text{te}}, \boldsymbol{x}_\ell^{\text{te}})\kappa(\boldsymbol{x}_i^{\text{te}}, \boldsymbol{x}_{\ell'}^{\text{te}}) \\
&\quad + \frac{\alpha}{n_{\text{tr}}}\sum_{j=1}^{n_{\text{tr}}} \kappa(\boldsymbol{x}_j^{\text{tr}}, \boldsymbol{x}_\ell^{\text{te}})\kappa(\boldsymbol{x}_j^{\text{tr}}, \boldsymbol{x}_{\ell'}^{\text{te}});
\end{aligned}$$

$\widehat{\boldsymbol{h}}$ is the n_{te}-dimensional vector with the ℓ-th element $\widehat{h}_\ell = \frac{1}{n_{\text{te}}}\sum_{i=1}^{n_{\text{te}}} \kappa(\boldsymbol{x}_i^{\text{te}}, \boldsymbol{x}_\ell^{\text{te}})$. Then the solution to Eq. (5) can be *analytically* obtained as

$$\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{H}} + \nu\boldsymbol{I})^{-1}\widehat{\boldsymbol{h}}, \tag{6}$$

where $\boldsymbol{I}$ is the $n_{\text{te}} \times n_{\text{te}}$-dimensional identity matrix.

The performance of RuLSIF depends on the choice of the kernel bandwidth τ and the regularization parameter ν. Model selection of RuLSIF is possible based on cross-validation with respect to the squared-error criterion J (Yamada *et al.*, 2013).

3.2 Illustrative example

We illustrate the efficacy of the proposed USDA approach in Figure 2 on a simple synthetically generated example. For this illustration, as well as for remainder of the paper we use USDA as part of the Twin Gaussian Regression (TGP) model (we review TGP and discuss how it can be learned with importance weighting imposed on the samples in Section 5 and Appendix A), which is able to deal with non-linear multi-modal relationships between input and output variables. Note, that for value of input close to $x = 0.5$, the output can have up to 3 modes. The proposed USDA-TGP approach ($\alpha = 0.5$) can substantially improve on the mean squared error with respect to the ground truth.

3.3 Issues with USDA under Severe Biases

The proposed USDA is well suited for reducing effects of sampling bias (as is illustrated in Figure 2), but the approach is not well suited for dealing with more severe biases (e.g., those that induce structured changes between the training and test sets) and therefore make the training and test domains largely disjoint (e.g., see Figure 3 (a)). This is easy to see, as the relative importance weight, in Eq.(3), goes to a constant $1/(1-\alpha)$ under such scenario. As a result, the unsupervised approach becomes ineffective and potentially large benefits can be obtained by leveraging small amounts of labeled data in the target domain, leading to the semi-supervised domain adaptation we introduce next.

4 Semi-Supervised Domain Adaptation (SSDA) for Regression

Here, we propose a semi-supervised domain adaptation method (SSDA) for 3D pose estimation to deal with more severe biases. Our semi-supervised approach is a generalization of USDA introduced in the previous section, and amounts to first projecting training and test samples to a higher dimensional space, through feature augmentation, then applying re-weighting to training and labeled test instances based on the ratio of their probabilities under the test and training marginals.

Let $\mathcal{X}^{\mathrm{tr}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of training image feature vector $\boldsymbol{x}^{\mathrm{tr}}$, $\mathcal{Y}^{\mathrm{tr}}(\subseteq \mathbb{R}^{d_{\mathrm{y}}})$ be the domain of training pose vector $\boldsymbol{y}^{\mathrm{tr}}$, $\mathcal{X}^{\mathrm{te}}(\subseteq \mathbb{R}^{d_{\mathrm{x}}})$ be the domain of test image feature vector $\boldsymbol{x}^{\mathrm{te}}$, and $\mathcal{Y}^{\mathrm{te}}(\subseteq \mathbb{R}^{d_{\mathrm{y}}})$ be the domain of test pose vector $\boldsymbol{y}^{\mathrm{te}}$. Suppose we are given n_{tr} and n'_{te} i.i.d. training and test image-pose feature pairs and $n_{\mathrm{te}} - n'_{\mathrm{te}}$ i.i.d. test image feature vectors,

$$\begin{aligned}
&\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}}) \mid \boldsymbol{x}_i^{\mathrm{tr}} \in \mathcal{X}^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}} \in \mathcal{Y}^{\mathrm{tr}},\ i = 1, \ldots, n_{\mathrm{tr}}\},\\
&\{(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}}) \mid \boldsymbol{x}_j^{\mathrm{te}} \in \mathcal{X}^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}} \in \mathcal{Y}^{\mathrm{te}},\ j = 1, \ldots, n'_{\mathrm{te}}\},\\
&\{\boldsymbol{x}_j^{\mathrm{te}} \mid \boldsymbol{x}_j^{\mathrm{te}} \in \mathcal{X}^{\mathrm{te}},\ j = n'_{\mathrm{te}}, \ldots, n_{\mathrm{te}}\},
\end{aligned}$$

drawn from distributions with densities $p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$, $p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y})$, and $p_{\mathrm{te}}(\boldsymbol{x})$ respectively; note, $n'_{\mathrm{te}} \ll n_{\mathrm{te}}$. Since we have n'_{te} test image-pose feature pairs, it is natural to include them in the training data set. Thus, we use $\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \cup \{(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}})\}_{j=1}^{n'_{\mathrm{te}}}$ as a new traning data set and assume the samples are drawn from a distribution with the density $p'_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$.

The final goal of SSDA for 3D pose estimation is to learn a function $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})$ with low expected pose error in the target domain based on the training and test image-pose feature pairs and test image feature vectors.

Similar to the USDA formulation in Section 3, learning of parameters amounts to solving the following optimization problem:

$$\min_{\boldsymbol{\Theta}} \left[\iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})) p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \right], \tag{7}$$

where $\boldsymbol{\Theta}$ are the model parameters.

However, because we can have a small number of test image-pose feature pairs, the optimization in Eq.(7) is rather difficult. Thus, we again consider the following covariate shift adaptation problem (Shimodaira, 2000):

$$\min_{\boldsymbol{\Theta}} \left[\iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{z}; \boldsymbol{\Theta})) w(\boldsymbol{z}) p'_{\mathrm{tr}}(\boldsymbol{z}, \boldsymbol{y}) \mathrm{d}\boldsymbol{z} \mathrm{d}\boldsymbol{y} \right], \tag{8}$$

where $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x})$ is the transformed sample of $\boldsymbol{x}$, $\boldsymbol{g}(\cdot)$ is an arbitrarry transformation function, and $w(\boldsymbol{z}) = \frac{p_{\mathrm{te}}(\boldsymbol{z})}{p'_{\mathrm{tr}}(\boldsymbol{z})}$ is the *importance weight* function. The difference from the USDA formulation is that we use feature transformation (i.e., $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x})$).

The advantage of the proposed SSDA approach is two fold: (1) the feature transformation can reduce the effect of non-overlapping regions between training and testing distributions (making the covariate shift assumption[3] valid), and (2) we can reduce sample selection bias in domain adaptation with the use of large number of test image features through covariate shift adaptation. Our proposed method consists of the following three steps that we describe in turn:

(i) Feature transformation $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x})$ obtained using feature augmentation (see Section 4.1).

(ii) Estimation of the importance weights $w(\boldsymbol{z})$ using RuLSIF (see Section 3.1).

(iii) Importance weighted learning of parameters $\boldsymbol{\Theta}$ of $\boldsymbol{f}(\boldsymbol{z}; \boldsymbol{\Theta})$ from weighted augmented feature-pose pairs (see Section 5 and Appendix A).

4.1 Feature Transformation

We adopt the supervised domain adaptation method called *EasyAdapt* (EA) (Daumé, 2007), which proved useful in natural language processing (NLP) community. In this paper, we further extend the EA method by introducing a new parameter β $(0 \leq \beta \leq 1)$ which controls the adaptiveness to the target data[4]:

$$\boldsymbol{z}^{\mathrm{tr}} = [\boldsymbol{x}^{\mathrm{tr}\top} \;\; \beta\boldsymbol{x}^{\mathrm{tr}\top} \;\; \mathbf{0}_{\mathrm{d}}{}^{\top}]^{\top},$$
$$\boldsymbol{z}^{\mathrm{te}} = [\boldsymbol{x}^{\mathrm{te}\top} \;\; \mathbf{0}_{\mathrm{d}}{}^{\top} \;\; \beta\boldsymbol{x}^{\mathrm{te}\top}]^{\top},$$

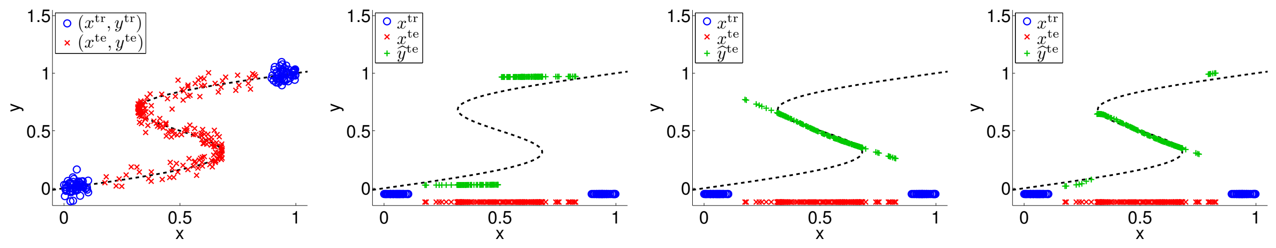
where $\mathbf{0}_{\mathrm{d}}$ is a d-dimensional vector with all zeros. Intuitively, these transformations map the original feature vectors into three versions: a *general* version, a *source-specific* version and a *target-specific* version. Note, $\beta = 0$ gives no adaptation in the sense of EA, while $\beta = 1$ gives the same effect as the original EA (Daumé, 2007); $0 < \beta < 1$ will give an intermediate adaptation. Moreover, setting $\beta = 0$ corresponds to simply merging $\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{y}_j^{\mathrm{te}})\}_{j=1}^{n'_{\mathrm{te}}}$ and regarding it as the training dataset. As such, the proposed SSDA formulation is a generalization of the original feature augmentation approach in Daumé (2007). In addition, this simple extension allows us to tune the adaptiveness based on prior information or using cross validation, which is desirable.

Analysis and intuition: To analyze the behavior of this transformation, let us focus on a Gaussian kernel for $\boldsymbol{x}$: $\kappa(\boldsymbol{x}, \boldsymbol{x}'; \rho^2) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\rho^2}\right)$. Then, when $\boldsymbol{x}$ and $\boldsymbol{x}'$ are samples from the same domain, we can compute the Gaussian kernel between $\boldsymbol{z}$ and $\boldsymbol{z}'$ as

$$K(\boldsymbol{z}, \boldsymbol{z}') = \kappa(\boldsymbol{x}, \boldsymbol{x}'; \rho^2)\kappa(\boldsymbol{x}, \boldsymbol{x}'; \frac{\rho^2}{\beta^2}).$$

[3]Covariate shift assumption formally amounts to assuming that conditional distributions on the source and target domains are the same but the marginal distributions are different.

[4]While it is possible to set $\beta > 1$, this gives an even higher importance to the *target* domain samples (meanwhile largely ignoring contributions from the *source* domain samples), which with few *target* samples leads to overfitting.



(a) Data samples (ROD = 0.485)
(b) USDA-TGP Source (MSE=0.248)
(c) USDA-TGP Target (MSE=0.079)
(d) SSDA-TGP (MSE=0.054)

Figure 3: Predicted outputs y using USDA-TGP with source data samples (b), USDA-TGP with target data samples (c), and SSDA-TGP method (d) in green. Samples from the model $x = y + 0.3\sin(2\pi y) + e$ where $e \sim \mathcal{N}(0, 0.05^2)$; $\circ$ and $\times$ are training and test samples respectively (for clarity we also illustrate marginals $p_{\mathrm{tr}}(\boldsymbol{x})$ and $p_{\mathrm{te}}(\boldsymbol{x})$ in (b), (c), and (d) bottom). Note that the input-output test samples are not used in the training of USDA-TGP Source, and we used only 100 input-output test samples for USDA-TGP Target and the proposed SSDA-TGP method.

On the other hand, when the domain is not the same,

$$\begin{aligned}K(\boldsymbol{z}, \boldsymbol{z}') &= \kappa(\boldsymbol{x}, \boldsymbol{x}'; \rho^2)\kappa(\boldsymbol{x}, \boldsymbol{x}'; \frac{\rho^2}{\beta^2}) \exp\left(-\frac{2\beta^2 \boldsymbol{x}^\top \boldsymbol{x}'}{\rho^2}\right), \\ &\leq \kappa(\boldsymbol{x}, \boldsymbol{x}'; \rho^2)\kappa(\boldsymbol{x}, \boldsymbol{x}'; \frac{\rho^2}{\beta^2}),\end{aligned}$$

where the inequality holds when $\boldsymbol{z}$ and $\boldsymbol{z}'$ only take positive values such as HoG features. Thus, data samples from the same domain are enhanced compared to those from different domains. Intuitively, this means that the labeled data samples from the target domain have a larger influence, as controlled by β, than samples from the source domain when making predictions about target (test) data. A more complete analysis of this is given in Daumé (2007). Note, if elements of $\boldsymbol{x}$ take negative value, we can use a different type of feature augmentation. For example, instead of concatenating the features in the original space, as we do here, we can first map each of the three elements of the transformed feature vector into the kernel space and then concatenate them in the kernel space, which would allow us to apply the method for any features so long as we have a positive semi-definite kernel (Daumé, 2007). The approach also extends to problems with more than two domains.

4.2 Illustrative Example

We compare the performance of USDA to SSDA approach on a one dimensional synthetic example in Figure 3. Note, that the setting is similar to the one in Figure 2, with two notable differences: (i) the source and target distributions are more disjoint (Relative Pearson Divergence (ROD) = 0.485 while the data in Figure 2 (a) have ROD = 0.177; see Section 7) and (ii) we assume that a number (100 in this case) labeled input-output pairs are available in the target domain. Similarly to prior example, we compare USDA and SSDA in the context of Important Weighted Twin Gaussian Processes Regression which we discuss at length in the next section and Appendix A.

It is clear that USDA-TGP ($\alpha = 0.5$) performs poorly regardless whether source or labeled target data is used for training. We note that USDA-TGP trained on the labeled target samples performs better here because we have relatively many labeled target samples – 100 in this case. In contrast, SSDA-TGP, here with $\alpha = 0.5$ and $\beta = 0.5$, performs much better than both USDA alternatives with 32% lower mean squared error (MSE) over USDA-TGP Target and 78% lower MSE over USDA-TGP Source.

5 Importance Weighted Twin Gaussian Processes Regression

USDA and SSDA approaches introduced in previous sections are applicable to variety of regression models. The only restriction placed on the regression model, is that learning must take into account weighted samples. We discuss variety of alternatives in our earlier work (Yamada *et al.*, 2012); here we focus on twin Gaussian processes regression (Bo and Sminchisescu, 2010) which proved very effective for high dimensional problems with potentially structured outputs and multi-valued relationships between inputs and outputs. We overview the twin Gaussian processes (TGP) regression (Bo and Sminchisescu, 2010) and importance-weighted twin Gaussian process (IWTGP) regression (Yamada *et al.*, 2012) in Appendix A.

Computational complexity: Since TGP/IWTGP requires matrix inversions of $n_{\mathrm{tr}} \times n_{\mathrm{tr}}$ matrices, the complexity of solving Eq.(21) is $O(n_{\mathrm{tr}}^3)$, which is impractical when n_{tr} is large. To deal with this issue, a common solution, which we adopt, is to first find M nearest neighbors to a test input and estimate TGP on the reduced set of training paired samples. The inverse matrix in Eq.(21) can then be efficiently computed with complexity $O(M^3)$. However, in addition to matrix inversion, TGP needs to solve a nonlinear optimization problem, Eq.(21), which tends to be computationally expensive and typically requires L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) solver and computation of gradients.

6 Computationally Efficient Twin Gaussian Processes Regression

TGP and IWTGP, discussed in the previous section, require solving of a nonlinear optimization problem which tends to be computationally expensive. As a consequence, our domain adaptation variants, USDA-TGP and SSDA-TGP inherit these computational burdens. To alleviate these issues, we propose computationally efficient alternative to TGP, which we call Direct TGP (dTGP), and its importance weighted variants.

To cope with computational challenges in TGP, we propose an approximation which results in a model similar to locally weighted K-nearest neighbor regression (WKNN). In this model the weights for the samples are estimated such that the Kullback-Leibler divergence between input and output Gaussian distributions is minimized, approximating the original TGP objective. This is efficient, as the weights can be analytically obtained. We can then estimate an (output) pose as a weighted sum of K-nearest training pose vectors, where top K nearest training pose vectors are obtained based on the estimated weighting.

6.1 Direct Twin Gaussian Processes (dTGP)

A computational limitation of TGP is that it needs to optimize $\boldsymbol{y}$ variable inside a Gaussian kernel, which makes TGP optimization nonlinear. To avoid solving nonlinear optimization problem, we adopt two-step approach which is similar to weighted K-nearest neighbor regression (WKNN) (Shakhnarovich *et al.*, 2003). More specifically, since $\boldsymbol{l}(\boldsymbol{y}) = [L(\boldsymbol{y},\boldsymbol{y}_1),\ldots,L(\boldsymbol{y},\boldsymbol{y}_n)]^\top$ can be regarded as a sample re-weighting vector, we first estimate $\boldsymbol{l}_{\boldsymbol{y}}$ and choose top K nearest output vectors $\{\boldsymbol{y}'_k\}_{k=1}^K$ by ranking based on $\boldsymbol{l}_{\boldsymbol{y}}$, and then, second, estimate an output $\boldsymbol{y}$ as a weighted sum over K-nearest neighbors. Therefore, the key issue is efficient estimation of $\boldsymbol{l}(\boldsymbol{y})$.

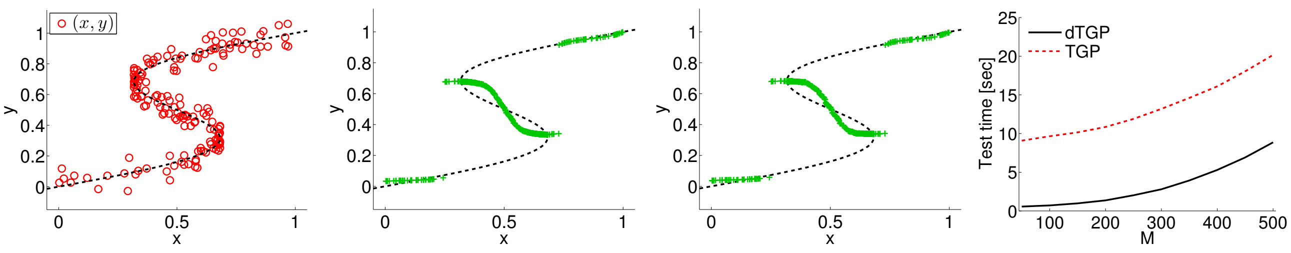
Estimating re-weighting vector: We regard $\boldsymbol{l}_{\boldsymbol{y}} = \boldsymbol{l}(\boldsymbol{y})$ as a variable to be optimized. Then, the optimization problem in Eq.(21) can be expressed as

$$\begin{aligned}\min_{\boldsymbol{l}_{\boldsymbol{y}}} \quad & \left[1+\lambda_{\mathrm{y}} - 2\boldsymbol{l}_{\boldsymbol{y}}^\top \boldsymbol{u} - \eta \log\left[1+\lambda_{\mathrm{y}} - \boldsymbol{l}_{\boldsymbol{y}}^\top \boldsymbol{L}^{-1}\boldsymbol{l}_{\boldsymbol{y}}\right]\right] \\ \text{s.t.} \quad & 0 \le l_{\boldsymbol{y},i} \le 1+\lambda_{\mathrm{y}},\ i=1,\ldots,n, \end{aligned} \tag{9}$$

where we use $L(\boldsymbol{y},\boldsymbol{y}) = 1+\lambda_{\mathrm{y}}$. Note, to further speed up the inference, we first estimate $\boldsymbol{l}_{\boldsymbol{y}}$ without box constraint and then clamp $\boldsymbol{l}_{\boldsymbol{y}}$ to satisfy the constraint.

Taking derivative of the objective function in Eq.(9) with respect to $\boldsymbol{l}_{\boldsymbol{y}}$ and equating it to zero, we get

$$\boldsymbol{l}_{\boldsymbol{y}} = \mu \boldsymbol{L}\boldsymbol{u}, \tag{10}$$



(a) Data samples (b) TGP (MSE=0.0552) (c) dTGP (D) (MSE=0.0553) (d) Computational Time

Figure 4: Predicted outputs y by TGP (b) and dTGP (c) for multi-modal data, where we set M = 200. (a): Samples from the model $x = y + 0.3\sin(2\pi y) + e$ where $e \sim \mathcal{N}(0, 0.05^2)$. (d): Average test time for TGP and dTGP over 500 test samples with respect to the number of nearest neighbors M.

where $\mu = \frac{(1+\lambda_{\rm y} - \boldsymbol{l}_{\boldsymbol{y}} \boldsymbol{L}^{-1} \boldsymbol{l}_{\boldsymbol{y}})}{\eta}$ is a scalar. By plugging Eq. (10) back into Eq. (9), we can rewrite the optimization problem as

$$\min_{\mu} \quad \Big[1 + \lambda_{\rm y} - 2\mu a - \eta \log \left[1 + \lambda_{\rm y} - \mu^2 a\right]\Big], \tag{11}$$

where $a = \boldsymbol{u}^\top \boldsymbol{L} \boldsymbol{u}$ is a constant. Note, we only need to estimate a scalar μ, while originally we needed to optimize $\boldsymbol{y} \in \mathbb{R}^{d_{\rm y}}$. Hence the new optimization problem is much easier to solve compared to the original optimization problem.

Taking derivative of Eq.(11) with respect to μ and equating it to zero, we get the following analytical solution:

$$\widehat{\mu} = \frac{-\eta + \sqrt{\eta^2 + 4a(1 + \lambda_{\rm y})}}{2a}.$$

The final solution can be written as

$$\widehat{\boldsymbol{l}}_{\boldsymbol{y}} = \min((1 + \lambda_{\rm y})\mathbf{1}_n, \max(\mathbf{0}_n, \widehat{\mu}\boldsymbol{L}\boldsymbol{u})),$$

where $\mathbf{1}_n$ denotes the n-dimensional vector with all ones, $\mathbf{0}_n$ denotes the n-dimensional vector with all zeros, and the 'max' and 'min' operation for vectors are applied in an element-wise manner.

Estimating Pose using Weighted K-nearest Neighbor Regression: We estimate $\boldsymbol{y}$ using weighted KNN regression (Shakhnarovich *et al.*, 2003). We first obtain K nearest neighbors and their weights $\{(\boldsymbol{y}'_k, \widehat{l}'_{\boldsymbol{y},k})\}_{k=1}^{K}$ from $\{\boldsymbol{y}_i\}_{i=1}^{n}$, by sorting training samples based on $\widehat{\boldsymbol{l}}_{\boldsymbol{y}}$, and estimate $\boldsymbol{y}$ as

$$\widehat{\boldsymbol{y}} = \sum_{k=1}^{K} \widehat{\gamma}_k \boldsymbol{y}'_k,$$

where $\widehat{\gamma}_k$ are the weights which are computed from $\widehat{\boldsymbol{l}}_{\boldsymbol{y}}$. In contrast to traditional WKNN where the weights are a function of the distance in the input space, in our case the weights are functions of both similarity of the test point to the training pairs, in the input space, and the similarity among the training pairs themselves, in the output space. We evaluate the following weighting functions:

Uniform: $\widehat{\gamma}_k = \frac{1}{K}, k = 1, \ldots, K$

Gaussian: $\widehat{\gamma}_k = \frac{\widehat{l}'_{\boldsymbol{y},k}}{\sum_{k=1}^{K} \widehat{l}'_{\boldsymbol{y},k}}, k = 1, \ldots, K$

Distance: $\widehat{\gamma}_k = \frac{-1/\log(\widehat{l}'_{\boldsymbol{y},k})}{\sum_{k=1}^{K} -1/\log(\widehat{l}'_{\boldsymbol{y},k})}, k = 1, \ldots, K$

Note, since $\widehat{l}'_{\boldsymbol{y},k}$ takes value of a Gaussian kernel, $\widehat{l}'_{\boldsymbol{y},k} = \exp(-\frac{\|\boldsymbol{y}-\boldsymbol{y}_k\|^2}{2\rho_y})$, the Distance weighting may be approximated by $\|\boldsymbol{y} - \boldsymbol{y}_k\|^2 \propto -\log(\widehat{l}'_{\boldsymbol{y},k})$.

The advantage of the proposed method over the existing TGP is that we can estimate the output $\boldsymbol{y}$ using simple linear algebra, while the original TGP needs to solve a nonlinear optimization problem which tends to be computationally expensive. In addition, the proposed approach does not depend on a specific nonlinear optimization solver, and is very easy to implement.

Relation to Standard Weighted K-nearest Neighbor Regression: Here, we show that a standard WKNN regression (Shakhnarovich *et al.*, 2003) is a special case of our proposed method.

Let us assume that input and output distributions are the same, i.e., $\boldsymbol{K} = \boldsymbol{L}$ and $\lambda_z = \lambda_y$. Then, the estimated $\boldsymbol{l_y}$ is given by

$$\widehat{\boldsymbol{l}}_{\boldsymbol{y}} = \widehat{\mu}\boldsymbol{K}\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{z}) = \boldsymbol{k}(\boldsymbol{z}),$$

where $\widehat{\mu} = 1$ and $\boldsymbol{k}_k(\boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{z}_k-\boldsymbol{z}\|^2}{2\rho_z^2}\right)$. Hence the Gaussian weighting parameter can be given as

$$\widehat{\gamma}_k = \frac{\exp\left(-\frac{\|\boldsymbol{z}_k-\boldsymbol{z}\|^2}{2\rho_z^2}\right)}{\sum_{k=1}^{K}\exp\left(-\frac{\|\boldsymbol{z}_k-\boldsymbol{z}\|^2}{2\rho_z^2}\right)}, \ k = 1, \ldots, K,$$

where $\{\boldsymbol{z}'_k\}_{k=1}^K$ is the K-nearest neighbor set. The resulting weighting parameter is the same as the weighting in standard WKNN regression. Therefore the proposed method, in particular with Gaussian weighting, reduces to standard WKNN when covariance matrices are the same (i.e., $\boldsymbol{K} = \boldsymbol{L}$).

6.2 Direct Importance Weighted Twin Gaussian Processes (dIWTGP)

The proposed speedup technique can also be applied for importance weighted variant of TGP, called IWTGP after (Yamada *et al.*, 2012).

Estimating re-weighting vector: Again let us consider $\boldsymbol{l_y} = \boldsymbol{l}(\boldsymbol{y})$ as a variable to be optimized. Then, the optimization problem in Eq.(24) can be written as

$$\begin{aligned}&\min_{\boldsymbol{l_y}} \quad \left[1 + \lambda_{\mathrm{y}} - 2\boldsymbol{l}_{\boldsymbol{y}}^\top \boldsymbol{u}_{\boldsymbol{w}} - \eta_{\boldsymbol{w}} \log\left[1 + \lambda_{\mathrm{y}} - \boldsymbol{l}_{\boldsymbol{y}}^\top \boldsymbol{L}_{\boldsymbol{w}}^{-1} \boldsymbol{l}_{\boldsymbol{y}}\right]\right] \\ &\text{s.t.} \quad 0 \leq l_{\boldsymbol{y},i} \leq 1 + \lambda_{\mathrm{y}}, \ i = 1, \ldots, n. \end{aligned} \tag{12}$$

Taking derivative of Eq.(12) and equating it to zero, we can similarly obtain the sample re-weighting vector:

$$\widehat{\boldsymbol{l}}_{\boldsymbol{y}} = \min((1 + \lambda_{\mathrm{y}})\mathbf{1}_n, \max(\mathbf{0}_n, \widehat{\mu}_w(\boldsymbol{L}_{\boldsymbol{w}})\boldsymbol{u}_w)), \tag{13}$$

where $\widehat{\mu} = \frac{-\eta_w + \sqrt{\eta_w^2 + 4a_w(1+\lambda_{\mathrm{y}})}}{2a_w}$ and $a_w = \boldsymbol{u}_w^\top \boldsymbol{L}_{\boldsymbol{w}} \boldsymbol{u}_w$. As before, $\boldsymbol{y}$ can be estimated by weighted K-nearest neighbor regression. We call the importance weighted variant of dTGP – dIWTGP.

6.3 Illustrative Example

Figure 4 illustrates predicted outputs y using TGP and dTGP (Dist) for multi-modal data: $x = y + 0.3\sin(2\pi y) + e$ where $e \sim \mathcal{N}(0, 0.05^2)$. In these experiments, we use 10,000 training samples and 500 test samples. As Figures 4 (b) and (c) clearly show, dTGP (Dist) performs very similarly to the original TGP implementation (Bo and Sminchisescu, 2010). In addition, Figure 4 (d) shows the test computational time for 500 test samples with respect to the number of nearest neighbors M. The computational speed of the proposed method is about 20 times faster than that of the original TGP when $M = 50$. The computational cost of the proposed method gradually increase when M increases. This is because dTGPs need to compute a matrix inverse and the computational cost of inverse becomes dominant when M is large. In practice, setting M to 200 works favorably in terms of both performance and computational time.

7 Domain Similarity Estimation using relative Pearson Divergence

Our leading assumption is that SSDA is more effective, than USDA, when difference between source and target domains is larger. To analyze the relationship between the similarity of domains and the proposed USDA and SSDA approaches, we propose an approach to estimate the rank of domain (ROD) measure (Gong *et al.*, 2012), which is useful to measure the difference between source and target dataset, based on the relative Pearson (PE) Divergence (Yamada *et al.*, 2013). We use the relative Pearson (PE) Divergence, since it can be estimated analytically and has good non-parametric convergence property (Yamada *et al.*, 2013).

Let P_{tr} and P_{te} be probability distributions of samples in $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, then the relative PE divergence is defined as (Yamada *et al.*, 2013)

$$\mathrm{PE}_\alpha(P_{\mathrm{tr}} \| P_{\mathrm{te}}) = \int (w_\alpha(\boldsymbol{x}) - 1)^2 \, q_\alpha(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{14}$$

where $w_\alpha(\boldsymbol{x})$ is the relative density-ratio (a.k.a., relative importance weight in Eq.(3)) and $q_\alpha(\boldsymbol{x}) = (1-\alpha)p_{\mathrm{te}}(\boldsymbol{x}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x})$. The relative PE divergence is a squared loss variant of the KL divergence, and it takes non negative value and vanish when $P_{\mathrm{tr}} = P_{\mathrm{te}}$ (Ali and Silvey, 1966). We use the symmetrized relative Pearson divergence as the rank of domain estimation:

$$\mathrm{ROD}(P_{\mathrm{tr}}, P_{\mathrm{te}}) = \frac{1}{2}(\mathrm{PE}_\alpha(P_{\mathrm{tr}} \| P_{\mathrm{te}}) + \mathrm{PE}_\alpha(P_{\mathrm{te}} \| P_{\mathrm{tr}})).$$

Estimation of the Relative Pearson Divergence: Using estimator of the relative density-ratio $w_\alpha(\boldsymbol{x})$, which is efficiently computed using RuLSIF, we can construct estimator of the relative PE divergence (14). After a few lines of calculation, we can show that the relative PE divergence (14) is equivalently written as

$$\begin{aligned}\mathrm{PE}_\alpha(P_{\mathrm{tr}} \| P_{\mathrm{te}}) = &-\frac{\alpha}{2} \int w_\alpha(\boldsymbol{x})^2 p_{\mathrm{tr}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &- \frac{(1-\alpha)}{2} \int w_\alpha(\boldsymbol{x})^2 p_{\mathrm{te}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int w_\alpha(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.\end{aligned} \tag{15}$$

Based on this expression, we obtain the estimator of the relative PE divergence as

$$\begin{aligned}\widehat{\mathrm{PE}}_\alpha(P_{\mathrm{tr}} \| P_{\mathrm{te}}) = &-\frac{\alpha}{2n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \widehat{w}_\alpha(\boldsymbol{x}_i^{\mathrm{tr}})^2 \\ &- \frac{(1-\alpha)}{2n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} \widehat{w}_\alpha(\boldsymbol{x}_j^{\mathrm{te}})^2 + \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} \widehat{w}_\alpha(\boldsymbol{x}_j^{\mathrm{te}}).\end{aligned} \tag{16}$$

Finally, the estimator of ROD measure based on the relative PE divergence is given by

$$\widehat{\mathrm{ROD}}(P_{\mathrm{tr}}, P_{\mathrm{te}}) = \frac{1}{2}(\widehat{\mathrm{PE}}_\alpha(P_{\mathrm{tr}} \| P_{\mathrm{te}}) + \widehat{\mathrm{PE}}_\alpha(P_{\mathrm{te}} \| P_{\mathrm{tr}})).$$

In this paper, we experimentally use $\alpha = 0.5$.

8 Experiments

8.1 Evaluation of USDA and SSDA

We start by exploring the ability of USDA and SSDA to battle various biases in different transfer learning settings. As in simple examples before, we analyze USDA and SSDA in the context of Importance Weighted Twin Gaussian Processes Regression. We apply USDA-TGP and SSDA-TGP to two problems on publicly available datasets: (i) 3D pose estimation from monocular images - on the HumanEva-I dataset and (ii) 3D head pose estimation from range data – on the ETH Face Pose Range Image Dataset (Breitenstein *et al.*, 2008). We further explore the benefits of USDA vs. SSDA as a function of the similarity between source and target domains on these real world datasets.

To do the latter, we compute an ROD score (based on the formulation in Section 7) for each setting. Lower value of ROD score corresponds to more similarity between source and target domains; higher value of ROD implies more dissimilarity between source and target domains. Experimentally, we observe that values of ROD close to 0.2 or less imply relatively small bias in the data (e.g., sample selection bias); values of ROD around 0.35 moderate bias and ROD ≥ 0.4 implies large structural biases in the data.

We compare the proposed **USDA-TGP** and **SSDA-TGP** methods to the following baselines:

TGP (S): TGP learned with training (source) image-pose pairs

TGP (T): TGP learned with test (target) image-pose pairs

TGP (S+T): TGP learned with training (source) image-pose pairs and test (target) image-pose pairs

TGP (EA): TGP learned with training (source) image-pose pairs and test (target) image-pose pairs in the augmented space

WKNN: Weighted k nearest neighbor regression with $k = 25$, using training (source) image-pose pairs.

Note, that, much like TGP, **USDA-TGP** can be trained with different slices of data, so we test the following alternatives:

USDA-TGP (S): IWTGP with training (source) image-pose pairs and test (target) image features for learning

USDA-TGP (T): IWTGP with test (target) image-pose pairs and test (target) image features for learning

USDA-TGP (S+T): IWTGP with training (source) image-pose pairs, test (target) image-pose pairs and test (target) image features for learning

Further note, **TGP (S+T)**, **USDA-TGP (S+T)**, **TGP (EA)** are actually special cases of proposed **SSDA-TGP** model where ($\alpha = 0, \beta = 0$), ($\alpha = 0.5, \beta = 0$) and ($\alpha = 0, \beta = 1.0$) respectively.

8.1.1 3D Human Pose Estimation

For these experiments we utilize HumanEva-I dataset (Sigal and Black, 2006). HumanEva-I contains synchronized multi-view video and Mocap data. It consists of 3 subjects performing multiple activities: walking, jogging, boxing, throw and catch, and gesturing. We use the histogram of oriented gradient (HoG) features ($\in \mathbb{R}^{270}$) proposed in Bo and Sminchisescu (2010) (we refer to (Bo and Sminchisescu, 2010) for details). We use training and validations sub-sets of HumanEva-I and only utilize data from 3 color cameras with a total of 9630 image-pose frames for each camera. This is consistent with experiments in Bo and Sminchisescu (2010) and Yamada *et al.* (2012). We first divide 9630 image-pose frames into training and test data set. Then, we randomly sub-sample n'_{te} from the full test data set as *labeled* samples and used rest of test, $n_{\text{te}} - n'_{\text{te}}$ data samples, for testing. We randomly sub-sample n'_{te} image-pose pairs from the full test set; to alleviate the sampling bias we sample 100 times, learn 100 different models, and average their corresponding errors.

We test three transfer scenarios: (1) **subject transfer** bias (sampling bias) – the training data includes 2 subjects and test data comes from a 3-rd subject not used for training, (2) **motion transfer** bias (data sharing)

– the training data does not include test motion, and (3) **camera transfer** – camera view is different between training and test. We propose 3 experiments:

Subject transfer (C1-C3): Test subject is not included in training phase. Data from cameras C1, C2, and C3 is used for this experiment as independent single-view samples.

Camera transfer (C1): Camera 1 data is used for training and Camera 2 data is used for testing.

Motion transfer (C1-3): *Walk*, *Jog*, and *ThrowCatch* motions are used for training and *Boxing* and *Gestures* are used for testing. Data from cameras C1, C2, and C3 is used for this experiment as independent single-view samples.

Error metric: In HumanEva-I pose is encoded by (20) 3D joint markers defined relative to the 'torsoDistal' joint in camera-centric coordinate frame, so $\boldsymbol{y} = [\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(20)}]^\top \in \mathbb{R}^{60}$ and $\boldsymbol{y}^{(i)} \in \mathbb{R}^3$. Error (in mm) for each pose is measured as average Euclidean distance: $Error_{pose}(\widehat{\boldsymbol{y}}, \boldsymbol{y}^*) = \frac{1}{20}\sum_{m=1}^{20} \|\widehat{\boldsymbol{y}}^{(m)} - \boldsymbol{y}^{*(m)}\|$, where $\widehat{\boldsymbol{y}}$ is an estimated pose vector, and $\boldsymbol{y}^*$ is a true pose vector (see Sigal and Black (2006) for details).

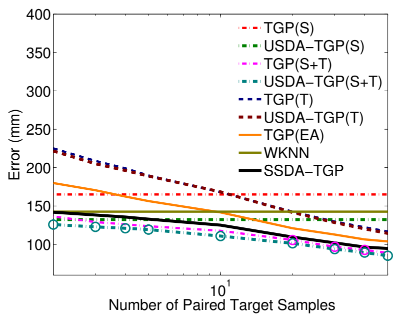
Parameters: For HumanEva-I dataset, we used the original parameter setting of Bo and Sminchisescu (2010): $\lambda_{\mathrm{z}} = \lambda_{\mathrm{y}} = 10^{-3}$, $2\rho_{\mathrm{z}}^2 = 5$, and $2\rho_{\mathrm{y}}^2 = 5 \times 10^5$. The number of M nearest neighbors in TGP and USDA-TGP is set to $min(300, n_{\mathrm{tr}})$. In USDA-TGP, we set $\alpha = 0.5$, and $b_{\mathrm{te}} = \min(500, n_{\mathrm{te}})$ to be consistent with original published results in Yamada *et al.* (2012). In SSDA-TGP, we choose the α and β parameters using cross-validation and let $b_{\mathrm{te}} = \min(500, n_{\mathrm{te}})$. In our cross-validation procedure we evaluate 9 different parameter settings where we set α and β to 0.0, 0.5, and 1.0 each and choose the parameter setting combination that maximizes performance on the validation set (in all experiments we use half of the labeled target samples for training and half for validation).

Performance and Analysis: Figures 5 (a)–(c) show the average mean pose estimation error as a function of the paired target set size (averaged over all motions and 100 runs of random sampling of paired target samples); Figure 5 (g) specifically shows performance with 5 labeled training samples, again averaged over 100 runs. The graphs and table show that the proposed USDA-TGP and SSDA-TGP outperform standard TGP on equivalent settings.

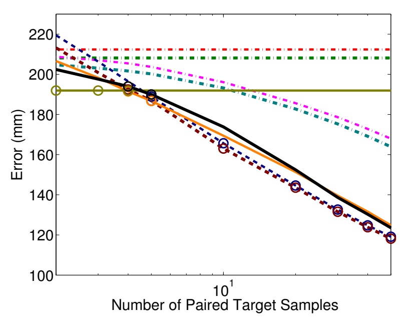
In particular, Figure 5 (g) shows that USDA-TGP outperforms TGP in all three transfer settings and on every single split of the data (S), (T) and (S+T). We notice that the largest boost (28% improvement in error) from USDA-TGP comes in the subject transfer case, with model trained on the source (S) samples. We believe there are two reasons for this. First, number of labeled target samples in this case is relatively small (only 5) in comparison to source samples, so training on them alone (T) is unlikely to be successful; adding them to a source samples (S+T) also has limited benefit. Second, the source and target domains are relatively similar (ROD = 0.207), making this setting particularly well suited for unsupervised domain adaptation. This also explains why SSDA-TGP performs marginally inferior to USDA-TGP on subject transfer.

Interestingly, however, USDA-TGP(S) starts to become much less effective as the dissimilarity between the target and source domains increases. In such cases, one can see that improvements obtained using a semi-supervised setting SSDA-TGP are considerably more pronounced; in Figure 5 (g) SSDA-TGP has 20% lower error than USDA-TGP (S+T) in the motion transfer setting with ROD= 0.406 (note, both approaches use the same exact labeled and unlabeled data). Further, for the intermediate ROD value of 0.353, in the camera transfer setting, the improvement of SSDA-TGP over USDA-TGP is also moderate at 5% lower error. This suggests that effectiveness of USDA and SSDA methods can be approximated by measuring the similarity/dissimilarity of the source and target domains.

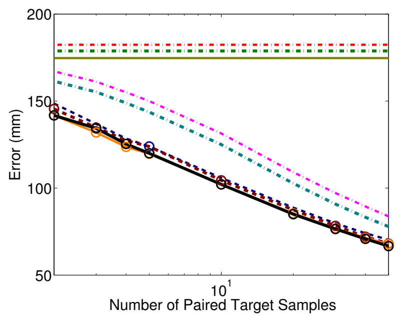
Figures 5 (d)–(f) show the comparison of SSDA-TGP with different α and β parameters. The automatically chosen parameters selected by cross-validation are labeled CV. We observed that setting $\beta = 0$ in SSDA-TGP tends to perform well when the training and test data set are not disjoint, while $\beta \geq 0.5$ tends to perform well when the training and test data set are disjoint (e.g., motion transfer). Cross-validation-based parameter selection tends to perform well when the number of labeled target samples is larger and becomes somewhat unstable when the number of labeled target samples is small. We observe that manually setting $\alpha = 0.5$ and $\beta = 0.5$ performs favorably in such cases and avoids instabilities of cross-validation. Paired t-tests were



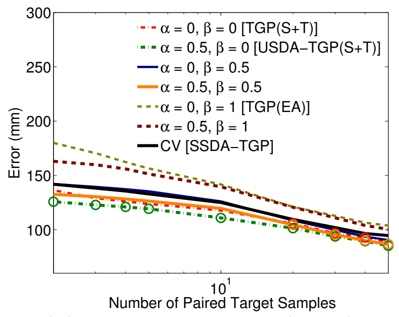
(a) Subject transfer (C1-3) (ROD = 0.207)



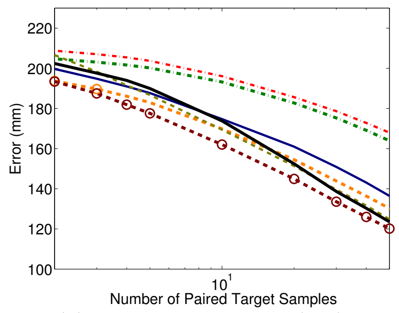
(b) Camera transfer (C1) (ROD = 0.353)



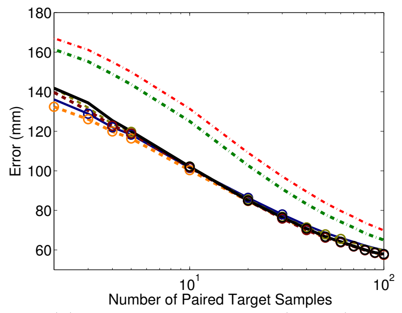
(c) Motion transfer (C1-3) (ROD = 0.406)



(d) Subject transfer (C1-3) (ROD = 0.207)



(e) Camera transfer (C1) (ROD = 0.353)



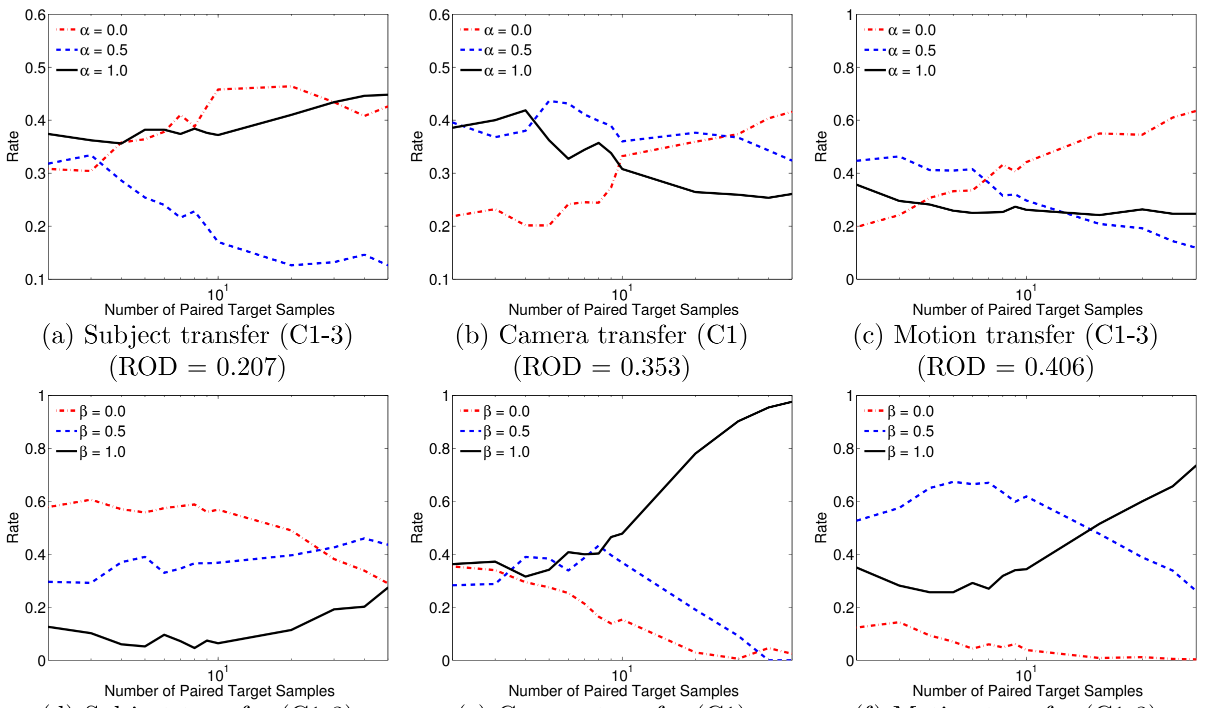
(f) Motion transfer (C1-3) (ROD = 0.406)

	Subject Transfer (ROD = 0.207)	Camera Transfer (ROD = 0.353)	Motion Transfer (ROD = 0.406)
TGP(S) (Bo and Sminchisescu, 2010)	165. 1	212.4	182.4
USDA-TGP(S)	132.4	208.2	178.8
TGP(S+T) (Bo and Sminchisescu, 2010)	123.1	203.7	150.0
USDA-TGP(S+T)	**119.3**	200.2	143.7
TGP(T) (Bo and Sminchisescu, 2010)	189.8	**188.9**	**123.9**
USDA-TGP(T)	189.0	**186.8**	**123.9**
TGP(EA)	156.4	**186.7**	**119.7**
WkNN	142.6	191.9	174.7
SSDA-TGP	132.9	**189.9**	**120.0**

(g) Quantitative results with 5 labeled paired target samples (based on (a), (b) and (c) above).

Figure 5: **Performance on** HUMANEVA-I dataset illustrated as a function of the number of paired target samples (a)-(c); we averaged the error over all motions for each subject and across (100) different random samplings of n'_{te}. Comparable methods according to the paired *t-test* at the significance level 5% are specified by '$\circ$'. Performance comparison of SSDA-TGP with respect to the α and β parameter illustrated in (d)-(f) as a function of the number of paired target samples; we averaged the error over all subjects. Comparable methods according to the paired *t-test* at the significance level 5% are specified by '$\circ$'. The table in (g) shows performance with 5 labeled paired target samples for easier analysis, and comparable methods according to the paired *t-test* are specified by bold font.

conducted for all experiments and we observe that SSDA-TGP statistically outperforms competitors at p=0.05 (5%) significance in most cases. We conducted the paired t-tests by first selecting the algorithm (parameter setting) with the lowest error and then performing pair-wise comparisons between it and every other algorithm



(a) Subject transfer (C1-3) (ROD = 0.207)
(b) Camera transfer (C1) (ROD = 0.353)
(c) Motion transfer (C1-3) (ROD = 0.406)
(d) Subject transfer (C1-3) (ROD = 0.207)
(e) Camera transfer (C1) (ROD = 0.353)
(f) Motion transfer (C1-3) (ROD = 0.406)

Figure 6: **Cross-validation parameter selection analysis on** HumanEva-I dataset illustrated as a function of the number of paired target samples; we averaged the error over all motions for each subject and across (100) different random samplings of n'_{te}. Each plot shows the fraction of the time a given parameter setting was chosen by cross-validation.

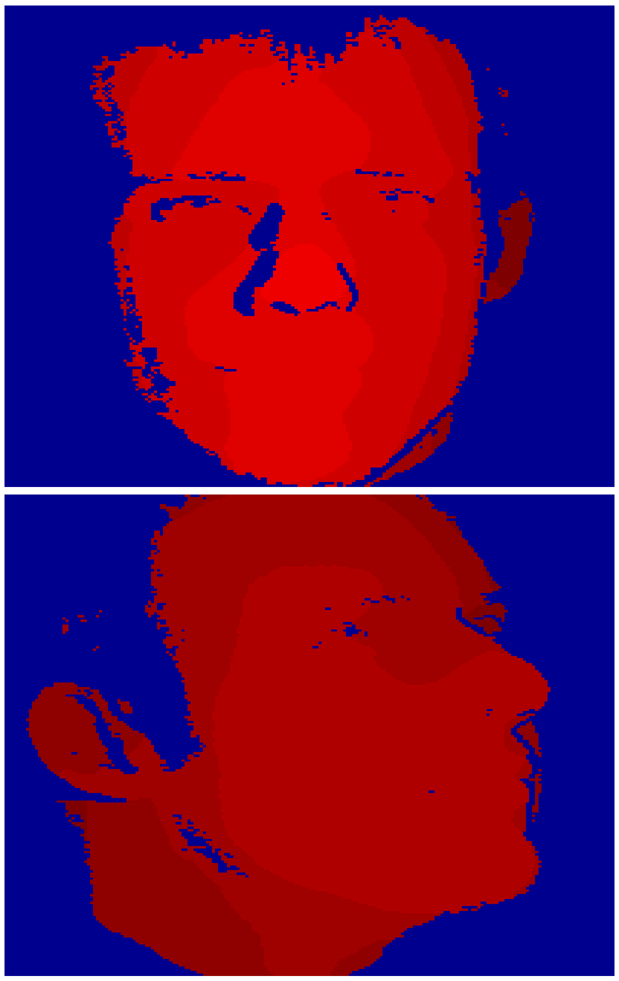
(parameter setting) considered.

One may be tempted to conclude that feature augmentation is what contributes the most to the performance, since in camera transfer and motion transfer settings performance of TGP(EA) is statistically indistinguishable from the proposed SSDA-TGP approach. However, we would like to highlight that in the subject transfer scenario SSDA-TGP does perform considerably (and statistically significantly) better than TGP(EA). In other words, we view SSDA-TGP as a more versatile approach that is capable of better dealing with variety of transfer settings.

The cross-validation parameter selection procedure is more closely evaluated in Figure 6 where we plot the fraction of runs (y-axis) in which cross-validation selected a given parameter value for α (a)-(c) and β (d)-(f) for a chosen number of labeled target samples (x-axis). As mentioned previously, we have multiple subjects and do 100 random samplings of n'_{te} for each transfer setting, so we have a distribution over the parameters chosen by cross-validation which we show in Figure 6. We observe that with few labeled target samples the results of cross-validation are not very stable and different parameters are chosen in many cases nearly equally frequently (Figure 6 (a,e)). As the number of labeled target samples increases cross-validation tends to become more stable with certain parameter setting being clearly preferred over the others (for example, see Figure 6 (c) or (e)).

8.1.2 3D Head Pose Estimation

We also assess the performance of the proposed USDA-TGP and SSDA-TGP methods on ETH Face Pose Range Image Dataset (Breitenstein *et al.*, 2008) (see Figure 1 (c)). The dataset contains 10780 range images of 20 people (3 females, 6 subjects recorded twice, with and without glasses) turning their head while captured at


(a) Depth Images

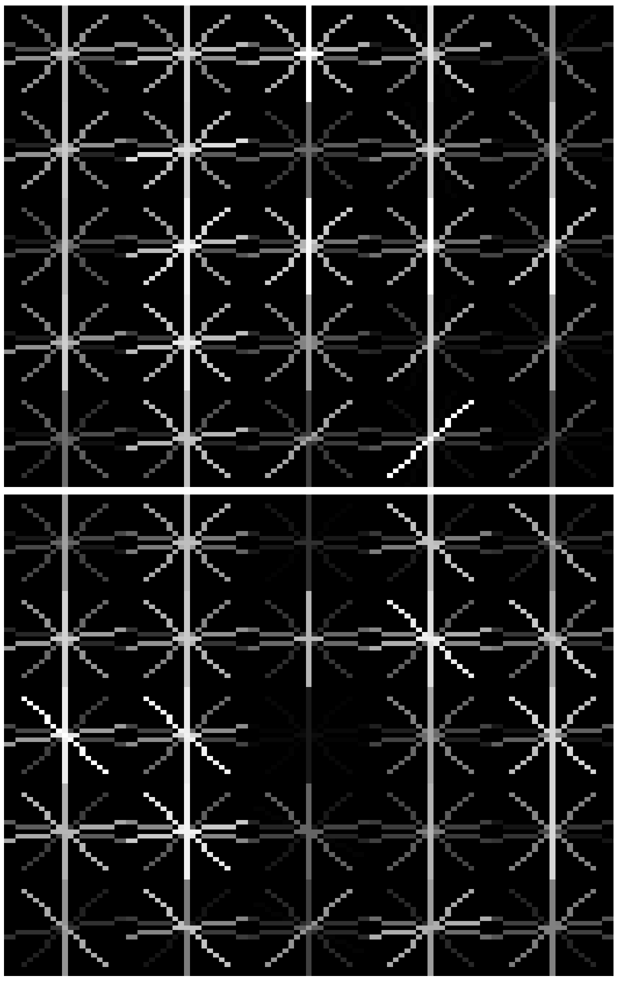

(b) HoG Descriptor

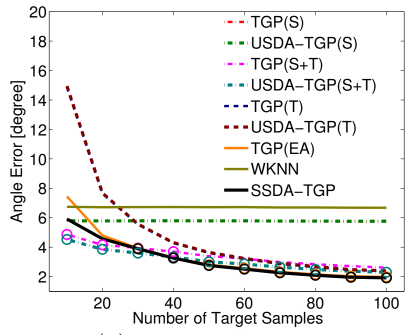
Figure 7: ETH Depth Face images and the corresponding HoG features.

28 fps by the range scanner of (Weise *et al.*, 2007). The resolution of each image is 640x480 pixels, and a face typically consists of 150x200 pixels. The head pose range is about $\pm 90°$ yaw and $\pm 45°$ pitch rotations. The provided ground truth for each image consists of the 3D nose tip coordinates and the coordinates of a vector pointing in the face direction (Breitenstein *et al.*, 2008).

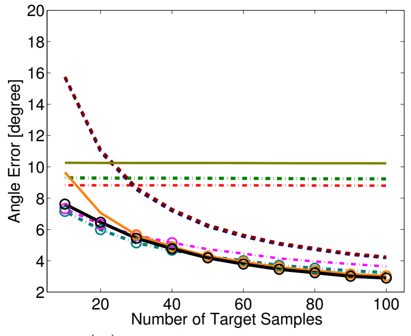
We compute HoG feature within a depth image's region of interest (ROI) (see Figure 7), in our case the bounding box around the face (200x200 pixels). To obtain the HoG, we divide the ROI into a non-overlapping 5×5 grid. Within each cell in the grid, we compute the orientation and magnitude of each pixel and obtain a 31 dimensional feature for each cell. We concatenated all features of cells to obtain 775 dimensional HoG feature ($\in \mathbb{R}^{775}$). Finally, we normalized the HoG vector to unit length. For pose vector $\boldsymbol{y}$, we use five dimensional vector comprising of yaw and pitch angle, and location of the nose ($\in \mathbb{R}^3$).

The bias in head pose estimation can come in (at least) two forms: the training (source) data may simply be biased and for example, not contain the subject present in the test (target) set, or an subject in test set wears glasses while no subjects wearing glasses were observed in the training set. Hence we propose 2 experiments:

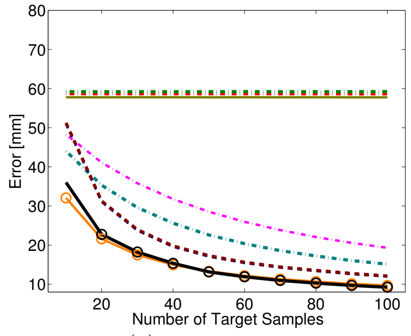
Subject transfer: Test subject is not included in the training set.



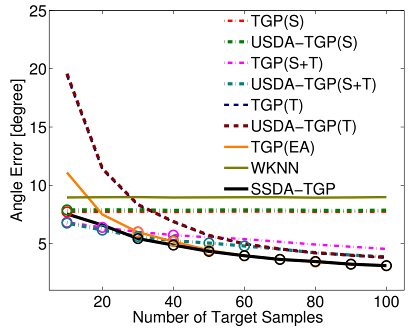
(a) Yaw angle error



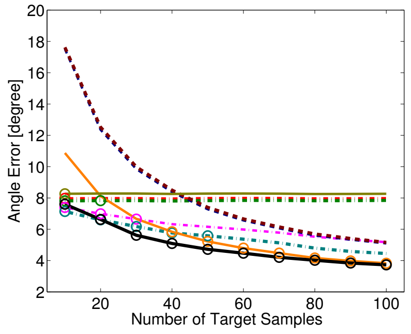
(b) Pitch angle error



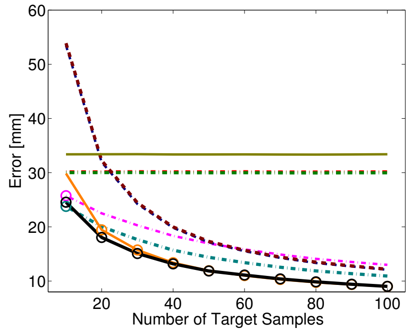
(c) Nose error



(d) Yaw angle error



(e) Pitch angle error



(f) Nose error

	Subject Transfer (ROD = 0.416)			Glass Transfer (ROD = 0.404)		
	Yaw [deg]	Pitch [deg]	Nose [mm]	Yaw [deg]	Pitch [deg]	Nose [mm]
TGP(S) (Bo and Sminchisescu, 2010)	5.78	8.82	58.6	7.74	7.99	30.25
USDA-TGP(S)	5.79	9.28	59.2	7.89	7.83	30.03
TGP(S+T) (Bo and Sminchisescu, 2010)	**3.95**	**5.62**	35.8	**6.00**	**6.65**	20.28
USDA-TGP(S+T)	**3.60**	**5.15**	29.6	**5.61**	**6.17**	17.65
TGP(T) (Bo and Sminchisescu, 2010)	5.54	8.53	23.7	8.30	9.85	24.22
USDA-TGP(T)	5.57	8.66	24.0	8.34	9.99	24.45
TGP(EA)	**3.93**	**5.66**	**17.5**	**5.98**	6.67	**15.74**
WkNN	6.72	10.24	57.8	8.98	8.28	33.41
SSDA-TGP	**3.89**	**5.48**	**18.2**	**5.42**	**5.61**	**15.08**

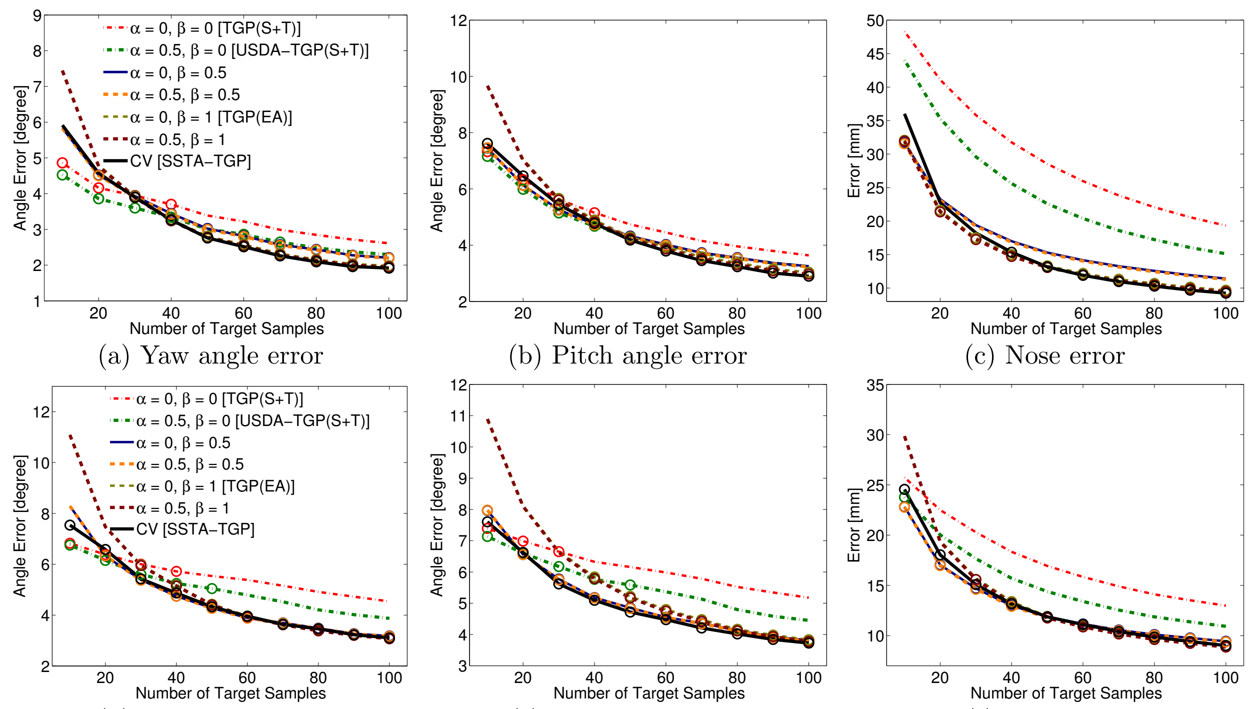
(g) Quantitative results with 30 labeled paired target samples (based on (a)–(f) above).

Figure 8: Performance on ETH Face Pose Range Image Dataset illustrated as a function of the number of test (target) samples; we averaged the error over all subjects. **Subject transfer (ROD = 0.416)**: (a)-(c) and **Glass transfer (ROD = 0.404):** (d)-(f). Comparable methods according to the paired *t-test* at 5% significance are specified by '○'. Results in (a)-(c) are competitive with (Breitenstein *et al.*, 2008), but are not directly comparable since the settings are different. The table in (g) shows performance with 30 labeled paired target samples for easier analysis, and comparable methods according to the paired *t-test* are specified by bold font.

Glass transfer: Subjects not wearing glass are used for training and subjects wearing glass are used for testing.

Error metric: We compute error in yaw (*degrees*), pitch (*degrees*), and nose position error (*mm*), where each error is measured as average Euclidean distance between estimated and ground truth data.

Parameters: For ETH data set, we experimentally (through grid search) set the TGP, USDA-TGP, and SSDA-



(a) Yaw angle error (b) Pitch angle error (c) Nose error

(d) Yaw angle error (e) Pitch angle error (f) Nose error

Figure 9: **Subject transfer (ROD = 0.415)**. (a)-(c): Performance comparison of SSDA-TGP with respect to the α and β parameter on the ETH face Pose Range Image dataset illustrated as a function of the number of paired target samples; we averaged the error over all subjects. **Glass transfer (ROD = 0.404)**. (d)-(f): Performance comparison of SSDA-TGP with respect to the α and β parameter on ETH face Pose Range Image dataset illustrated as a function of the number of paired target samples; we averaged the error over all subjects. Comparable methods according to the paired *t-test* at the significance level 5% are specified by '$\circ$'.

TGP parameters to $\lambda_{\mathrm{z}} = \lambda_{\mathrm{y}} = 10^{-5}$, $2\rho_{\mathrm{z}}^2 = 5 \times 10^3$, and $2\rho_{\mathrm{y}}^2 = 5 \times 10^7$. The number of M nearest neighbors in TGP and USDA-TGP is set to $min(300, n_{\mathrm{tr}})$. In SSDA-TGP, we choose the α and β parameters by cross-validation, and $b_{\mathrm{te}} = \min(500, n_{\mathrm{te}})$, respectively.

Performance and Analysis: Figure 9 shows the performance comparison of SSDA-TGP with respect to the α and β parameters on ETH face dataset. Figure 8 (g) shows performance with 30 labeled training samples. From these experiments, we observed that cross-validation (SSDA-TGP setting) and $\alpha = 0.5, \beta = 0.5$ both give favorable estimation accuracy with respect to the number of paired target samples. Moreover, Figures 8 (a)–(e) show the average mean yaw, pitch, and nose estimation error as a function of test set size (averaged over all subjects and 30 runs) for subject and glass transfer cases. The graphs clearly show that the proposed SSDA-TGP method outperforms non domain adaptation and USDA-TGP methods under most settings. Paired t-tests were conducted for all experiments. We observe that the USDA-TGP and SSDA-TGP methods statistically outperform competitors at p=0.05 (5%) significance. In both transfer sittings considered, the ROD values are relatively large, so SSDA-TGP tends to perform well; USDA-TGP (S+T) performs well on the yaw and pitch estimates in the subject transfer setting.

Figure 9 shows the performance of SSDA-TGP with different α and β parameters as compared to the cross-validation parameter setting procedure. The conclusions are largely similar to those drawn in the human pose experiments in the previous section. It is clear that cross-validation procedure outperforms fixed parameter values as number of labeled target samples increases.

8.2 Evaluation of Direct TGP

Now that we showed effectiveness of USDA and SSDA methods, we investigate the fast alternatives to the TGP that are able to improve on the speed of inference by a factor of 8 to 15 times. Since our fast approximation to TGP applies to all of the methods, instead of running a set of exhaustive experiments with variety of settings we explored in previous sections, here we focus on a few simple scenarios to illustrate overall effectiveness of the method.

8.2.1 HumanEva-I Data

We start by exploring performance of Direct TGP on HumanEva-I dataset under selection bias setting. Figure 10 shows the performance comparison with respect to M and K parameters for dTGP and dUSDA-TGP, respectively. We observe that performance of dTGP and dUSDA-TGP (similar to TGP and USDA-TGP) are insensitive to parameter M. The performance tends to asymptote once M is sufficiently large (> 200) and drops gracefully as M decreases. Some direct methods do appear sensitive to parameter K. In particular, one can see that with Uniform weighting the performance of dTGP and dUSDA-TGP actually degrades as K increases. This may seem counterintuitive at first, however, this can be easily explained. First, since Uniform weighting simply averages across nearest neighbors, having more neighbors implicitly means smoother and more regularized (and hence likely less accurate) prediction. Second, if regression is multi-modal, as is the case here, having more nearest neighbors with equal weighting will likely average across different modes (where as having much fewer will more likely focus on a single mode). Because Gaussian and Distance weightings take into account the distance between samples in both the input and output spaces both of these issues are mitigated. As a result, Gaussian and Distance variants of dTGP and dUSDA-TGP are relatively insensitive to K. Distance weighting performs best overall and tends to improve marginally as K increases.

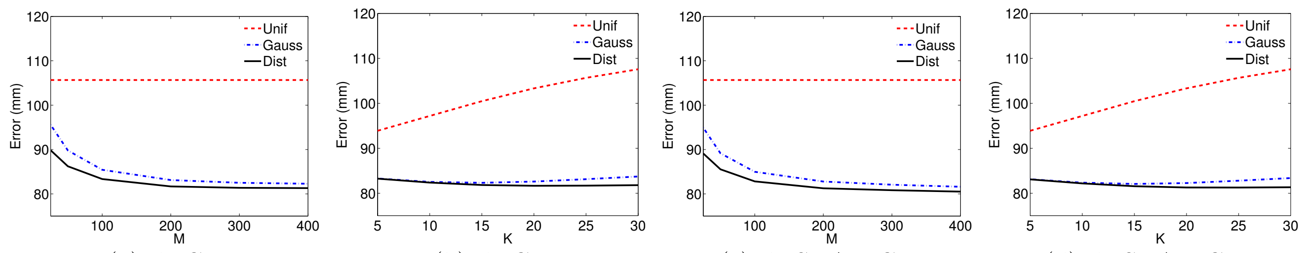
Based on results in Figure 10, we let $M = 200$ and $K = 25$ for the remaining experiments. Table 1 shows the pose estimation accuracy over the test set. Overall, the proposed direct method performs close to the original TGP and USDA-TGP, speeding them up by about 7 and 8 times, respectively. Distance weighting works the best among three considered weighting methods (less than 5% loss in performance on average). Note, when the number of training samples is big, the most computationally expensive operation is finding M-nearest neighbors. To deal with this issue, we can use *locality sensitive hashing* to further speed up computation (Shakhnarovich *et al.*, 2003).

8.2.2 Poser Data

We also notice that despite being an approximation, dTGP and dUSDA-TGP can actually perform better than the original in certain scenarios. We illustrate this by focusing on another public dataset made available by Agarwal and Triggs (2006). Poser dataset (Agarwal and Triggs, 2006) consists of 1927 training and 418 test images, which are synthetically generated, using Poser software package, from motion capture (Mocap) data (54 joint angles per frame). The image features, corresponding to bag-of-words representation with silhouette-based shape context features, and error metric are provided with the dataset. The ROD score of the Poser dataset is 0.220.

Error metric: The proposed error measure amounts to the root mean square error (in degrees), averaged over all joints angles, and is given by: $Error_{pose}(\widehat{\boldsymbol{y}}, \boldsymbol{y}^*) = \frac{1}{54} \sum_{m=1}^{54} \|(\widehat{\boldsymbol{y}}^{(m)} - \boldsymbol{y}^{*(m)}) \bmod 360°\|$, where $\widehat{\boldsymbol{y}} \in \mathbb{R}^{54}$ is an estimated pose vector, and $\boldsymbol{y}^* \in \mathbb{R}^{54}$ is a true pose vector (see Agarwal and Triggs (2006) for details).

Performance and Analysis: Table 2 shows the pose estimation result averaged over the test set. Proposed dTGPs and dUSDA-TGPs compares favorably with their conventional counterparts, speeding up performance by 11.5 and 10.6 times, respectively. Moreover, the Distance weighted K-nearest neighbor based dTGPs, dTGP(Dist) and dUSDA-TGP(Dist), outperform the original USDA-TGP in this data set.



(a) dTGP (b) dTGP (c) dUSDA-TGP (d) dUSDA-TGP

Figure 10: Performance comparison with respect to M and K parameters. (a,c): Performance comparison of dTGP and dUSDA-TGP with respect to M ($K = 25$). (b,d): Performance comparison of dTGP and dUSDA-TGP with respect to K ($M = 200$).

Table 1: Performance on the entire HumanEva-I dataset; we average errors over all motions and frames. The estimated ROD measure for this experiment is 0.207. The best method having the smallest error and comparable methods (error from a best method is less than $3mm$) are specified by bold face.

Subject		dTGP	dTGP	dTGP		dUSDA-TGP	dUSDA-TGP	dUSDA-TGP		
Train	Test	(Unif)	(Gauss)	(Dist)	TGP	(Unif)	(Gauss)	(Dist)	USDA-TGP	WKNN
S1,S2,S3	S1	97.1	88.8	**87.8**	**85.0**	97.1	88.7	87.6	**83.6**	94.3
S1,S2,S3	S2	105.1	79.4	78.1	**72.1**	105.1	79.0	77.6	**71.5**	100.7
S1,S2,S3	S3	117.1	80.6	**78.6**	**76.4**	117.1	80.2	**78.0**	**75.7**	110.4
Average error (mm)		106.4	82.9	81.5	77.8	106.4	82.6	81.0	76.9	101.8
Average test time (sec)		**3.46**	**3.46**	**3.46**	25.78	**3.75**	**3.75**	**3.75**	27.04	**0.76**

9 Conclusion

In this paper, we proposed a simple, yet effective, semi-supervised domain adaptation (SSDA) method for addressing training set bias in regression problems. Our SSDA approach is a generalization of the unsupervised domain adaptation (USDA), which we propose earlier in Yamada *et al.* (2012), and reduces to USDA when labeled data in the target domain is unavailable and/or under a particular setting of model parameters. Proposed SSDA, and the special case of USDA, are amenable to most discriminative and structured regression/prediction models and problems.

Using SSDA as the basis, we proposed a new form of Twin Gaussian Processes regression called SSDA-TGP. A key benefit of SSDA-TGP is that large structural biases in data can be alleviated by data sharing during learning. SSDA-TGP first projects the input image features into a higher dimensional space to alleviate domain biases caused by disjointness of training and test data sets. Importance weighted TGP (IWTGP) can then be used in this higher dimensional space to infer pose, while, in addition, removing the sample selection bias. We applied the proposed method to 3D head and 3D human pose estimation, and achieve state-of-the-art performance on standard datasets: HumanEva-I (Sigal and Black, 2006) and ETH Face Pose Range Image Dataset (Breitenstein *et al.*, 2008).

Further, we propose a measure of domain similarity, between the source and target domains, which allows us to explore tradeoffs between effectiveness of SSDA-TGP and it's unsupervised variant USDA-TGP. As a consequence, we are able to show that proposed SSDA-TGP model is more effective in removing large structural biases in data and in promoting data sharing during learning (which causes large dissimilarity between source and target domains). The unsupervised variant, USDA-TGP, on the other hand, is more effective in removing smaller biases such as selection bias.

Moreover, to speed up TGPs, we proposed a computationally efficient alternative to twin Gaussian processes, that we called direct TGP (dTGP). We show that dTGP, and proposed alternatives, are 8 to 15 times faster than traditional TGP formulation, with little to no loss in performance.

Table 2: Performance and computational training and test time of dTGP and dUSDA-TGP and existing methods on Poser dataset, where we set the number of nearest neighbors M as 200. Test time is measured over 418 images. The ROD score of the Poser data is 0.220. The best method having the smallest angle error is specified by bold face.

	dTGP (Unif)	dTGP (Gauss)	dTGP (Dist)	TGP	dUSDA-TGP (Unif)	dUSDA-TGP (Gauss)	dUSDA-TGP (Dist)	USDA-TGP	WKNN
Average error (deg)	5.77	5.45	**5.44**	5.69	5.77	5.44	**5.42**	5.67	5.69
Training time (sec)	0.15	0.15	0.15	0.15	1.22	1.22	1.22	1.22	–
Test time (sec)	**0.79**	**0.79**	**0.79**	12.32	**0.89**	**0.89**	**0.89**	12.63	0.51

Acknowledgement

The authors thank Dr. Liefeng Bo for providing us the TGP code and Dr. Michalis Raptis and Dr. Marek Vondrak for their valuable comments and contributions.

References

Agarwal, A. and Triggs, B. (2006). Recovering 3D human pose from monocular images. *IEEE Trans. on PAMI*, **28**(1), 44—58.

Ali, S. M. and Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society, Series B*, **28**, 131–142.

Bo, L. and Sminchisescu, C. (2010). Twin Gaussian processes for structured prediction. *IJCV*, **87**(1-2), 28–52.

Breitenstein, M., Kuettel, D., Weise, T., Van Gool, L., and Pfister, H. (2008). Real-time face pose estimation from single range images. In *CVPR*, pages 1–8.

Chen, Y., Kim, T.-K., and Cipolla, R. (2011). Silhouette-based object phenotype recognition using 3D shape priors. In *ICCV*, pages 25–32.

Daumé, H. (2007). Frustratingly easy domain adaptation. In *ACL*, pages 256–263.

Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. In *SIGKDD*, pages 109–117.

Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *CVPR*, pages 617–624.

Girshick, R., Shotton, J., Kohli, P., Criminisi, A., and Fitzgibbon, A. (2011). Efficient regression of general-activity human poses from depth images. In *ICCV*, pages 415–422.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073.

Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, pages 999–1006.

Jiang, J. (2007). A literature survey on domain adaptation of statistical classifiers.

Kanaujia, A., Sminchisescu, C., and Metaxas, D. (2007). Semi-supervised hierarchical models for 3D human pose reconstruction. In *CVPR*, pages 1–8.

Khosla, A., Zhou, T., Malisiewicz, T., Efros, A., and Torralba, A. (2012). Undoing the damage of dataset bias. In *ECCV*, pages 158–171.

Kulis, B., Saenko, K., and Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, pages 1785–1792.

Lim, J., Salakhutdinov, R., and Torralba, A. (2011). Transfer learning by borrowing examples for multi class object detection. In *NIPS*, pages 118–126.

Miller, E., Matsakis, N., and Viola, P. (2000). Learning from one example through shared densities of transforms. In *CVPR*, pages 464–471.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on Knowledge and Data Eng.*, **22**(10).

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *ECCV*, pages 213–226.

Shakhnarovich, G., Viola, P., and Darrell, T. (2003). Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750—757.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, **90**.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from a single depth image. In *CVPR*, pages 1297–1304.

Sigal, L. and Black, M. J. (2006). Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. In *TR CS-06-08, Brown Univ.*

Sigal, L., Balan, A., and Black, M. (2007). Combined discriminative and generative articulated pose and non-rigid shape estimation. In *NIPS*, pages 1337–1344.

Sminchisescu, C., Kanaujia, A., and Metaxas, D. (2006). Learning joint top-down and bottom-up processes for 3D visual inference. In *CVPR*, pages 1743–1752.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., and Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *NIPS*, pages 1433–1440.

Sun, M., Kohli, P., and Shotton, J. (2012). Conditional regression forests for human pose estimation. In *CVPR*, pages 3394–3401.

Torralba, A. and Efros, A. (2011). Unbiased look at dataset bias. In *CVPR*, pages 1521–1528.

Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multi-class object detection. In *CVPR*, pages 762–769.

Urtasun, R. and Darrell, T. (2008). Sparse probabilistic regression for activity-independent human pose inference. In *CVPR*, pages 1–8.

Weise, T., Leibe, B., and Van Gool, L. (2007). Fast 3D scanning with automatic motion compensation. In *CVPR*, pages 1–8.

Yamada, M., Sigal, L., and Raptis, M. (2012). No bias left behind: Covariate shift adaptation for discriminative 3D pose estimation. In *ECCV*, pages 674–687.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2013). Relative density-ratio estimation for robust distribution comparison. *Neural computation*, **25**(5), 1324–1370.

A Importance Weighted Twin Gaussian Processes Regression (IWTGP)

For completeness, we overview the importance-weighted variant of twin Gaussian processes (Bo and Sminchisescu, 2010) called IWTGP (Yamada *et al.*, 2012), which is a transfer learning method under *covariate shift* (Shimodaira, 2000).

Under covariate shift setup it is assumed that labeled training image-pose pairs $\{(\boldsymbol{z}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ drawn i.i.d. from $p(\boldsymbol{y}|\boldsymbol{z})p_{\mathrm{tr}}(\boldsymbol{z})$ and unlabeled test image features $\{\boldsymbol{z}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ drawn i.i.d. from $p_{\mathrm{te}}(\boldsymbol{z})$ (which is usually different from $p_{\mathrm{tr}}(\boldsymbol{z})$) are available. Note, for simplicity of notations, we regard $p_{\mathrm{tr}}(\boldsymbol{z},\boldsymbol{y})$ as $p'_{\mathrm{tr}}(\boldsymbol{z},\boldsymbol{y})$.

A.1 Gaussian Process Regression

We start by introducing standard Gaussian Process (GP) Regression. The GP regression assumes a linear model in the function space with Gaussian noise for the k-th dimension (e.g., joint position):

$$\boldsymbol{y}_k = f_k(\boldsymbol{z}) + e_k,\ e_k \sim \mathcal{N}(0,\sigma^2), \qquad f_k(\boldsymbol{z}) = \boldsymbol{b}_k^\top \boldsymbol{\phi}(\boldsymbol{z}), \tag{17}$$

where there is a zero mean Gaussian prior over the parameters $\boldsymbol{b}_k \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Sigma}_p)$; $\mathbf{0}_p$ is the p-dimensional zero vector and $\boldsymbol{\Sigma}_p$ is the p-dimensional covariance matrix, $\boldsymbol{\phi}(\boldsymbol{z})$ is the function which maps a $d_{\mathrm{z}} = 3d_{\mathrm{x}}$ dimensional input vector $\boldsymbol{z}$ into an p dimensional feature space. To make prediction for the test sample, one needs to average over all possible parameter values, weighted by their posterior, resulting in a Gaussian predictive distribution. GP has similar problems with multi-modality as kernel regression (KR). To address this limitation, TGP encodes the relations between both inputs and outputs using GP priors. This is achieved by minimizing the Kullback-Leibler divergence between the marginal GP of outputs (poses) and observations (features).

A.2 Twin Gaussian Processes Regression

In this section, we review the twin Gaussian processes regression (TGP), after (Bo and Sminchisescu, 2010), and point out potential computational issues with TGP.

Take an alternative view to GP regression, where a joint distribution over all training outputs, $\boldsymbol{Y} = [\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{n_{\mathrm{tr}}}] \in \mathbb{R}^{d_{\mathrm{y}}\times n_{\mathrm{tr}}}$, and an unknown test output $\boldsymbol{y}$ for a given corresponding input $\boldsymbol{z}$ takes the form of a joint Gaussian:

$$\begin{bmatrix} \boldsymbol{Y}_k^\top \\ y_k \end{bmatrix} \sim \mathcal{N}_{\boldsymbol{Z}}\left(\mathbf{0},\ \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}(\boldsymbol{z}) \\ \boldsymbol{k}(\boldsymbol{z})^\top & K(\boldsymbol{z},\boldsymbol{z}) \end{bmatrix}\right), \tag{18}$$

where $\boldsymbol{Y}_k$ is the k-th row of $\boldsymbol{Y}$, y_k is the k-th entry of $\boldsymbol{y}$, $\boldsymbol{K}$ is an $n_{\mathrm{tr}} \times n_{\mathrm{tr}}$ matrix with each element $\boldsymbol{K}_{ij} = cov(\boldsymbol{\phi}(\boldsymbol{z}_i), \boldsymbol{\phi}(\boldsymbol{z}_j))$ being a covariance function encoding correlations between pairs of random variables $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$; similarly $\boldsymbol{k}_i(\boldsymbol{z}) = cov(\boldsymbol{\phi}(\boldsymbol{z}_i), \boldsymbol{\phi}(\boldsymbol{z}))$ is a column vector of size $n_{\mathrm{tr}} \times 1$ and $K(\boldsymbol{z},\boldsymbol{z}) = cov(\boldsymbol{\phi}(\boldsymbol{z}), \boldsymbol{\phi}(\boldsymbol{z}))$. A popular choice for a covariance function is a Gaussian with noise:

$$\begin{aligned} cov(\boldsymbol{\phi}(\boldsymbol{z}_i), \boldsymbol{\phi}(\boldsymbol{z}_j)) &= K(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ &= \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{2\rho_{\mathrm{z}}^2}\right) + \lambda_{\mathrm{z}}\delta_{ij}, \end{aligned}$$

where, ρ_{z} is the kernel bandwidth parameter and λ_{z} is the noise variance; δ_{ij} is the Kronecker delta function.

Because the joint distribution is Gaussian, the predictive distribution is also Gaussian and can be obtained by conditioning on the observed training outputs $\boldsymbol{Y}_k$. The mean and variance of the predictive distribution can be derived in closed form:

$$\begin{aligned} m(y_k) &= \boldsymbol{Y}_k \boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{z}), \\ \sigma^2(y_k) &= K(\boldsymbol{z},\boldsymbol{z}) - \boldsymbol{k}(\boldsymbol{z})^\top \boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{z}). \end{aligned}$$

Unlike GP regression, TGP also defines the covariance function over outputs (not just inputs), which allows to model correlations in outputs. In addition, by minimizing KL divergence between the two Gaussian Processes (one going in the forward and one in the backwards direction) it's possible to focus on the most prominent mode in the potentially multi-modal mapping between image features and 3D pose.

To derive the backwards GP process, note that $\left[\boldsymbol{Y}_k^\top, y_k\right]^\top$, in Eq. (18), can also be thought of as a sample from a Gaussian distribution over the outputs,

$$\begin{bmatrix} \boldsymbol{Y}_k^\top \\ y_k \end{bmatrix} \sim \mathcal{N}_{\boldsymbol{Y}_k \cup y_k} \left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{L} & \boldsymbol{l}(y_k) \\ \boldsymbol{l}(y_k)^\top & L(y_k, y_k) \end{bmatrix} \right), \tag{19}$$

where we can empirically estimate covariance matrix as:

$$\boldsymbol{L}_{\boldsymbol{Y}_k \cup y_k} = \begin{bmatrix} \boldsymbol{Y}_k^\top \\ y_k \end{bmatrix} \begin{bmatrix} \boldsymbol{Y}_k & y_k \end{bmatrix}.$$

For a multivariate output case, we can treat each dimension k as an independent samples from the Gaussian distribution (see Eq. (18)) since covariance is independent of k. Hence, We can instead estimate covariance matrix as:

$$\boldsymbol{L}_{\boldsymbol{Y} \cup \boldsymbol{y}} = \frac{1}{d_{\mathrm{y}}} \begin{bmatrix} \boldsymbol{Y}^\top \\ \boldsymbol{y}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{Y} & \boldsymbol{y} \end{bmatrix}.$$

This, however, assumes that outputs along each dimension are i.i.d.; to account for the correlations between outputs, we can define a more general covariance function over the outputs, resulting in:

$$\begin{bmatrix} \boldsymbol{Y}^\top \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}_{\boldsymbol{Y} \cup \boldsymbol{y}} \left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{L} & \boldsymbol{l}(\boldsymbol{y}) \\ \boldsymbol{l}(\boldsymbol{y})^\top & L(\boldsymbol{y}, \boldsymbol{y}) \end{bmatrix} \right), \tag{20}$$

which bears close similarity to the original GP in Eq. (18).

TGP measures the offset between the true Gaussian distribution of the inputs, $\mathcal{N}_{\boldsymbol{Z}}$, and measured Gaussian distribution of the outputs, $\mathcal{N}_{\boldsymbol{Y} \cup \boldsymbol{y}}$, using Kullback-Leibler divergence. However, the output, $\boldsymbol{y}$, is unknown in this measure. To match the estimated output distribution and fully observed input one as much as possible, one is required to estimate output, $\hat{\boldsymbol{y}}$, by minimizing the Kullback-Leibler divergence:

$$\hat{\boldsymbol{y}} = \operatorname*{argmin}_{\boldsymbol{y} \in \mathbb{R}^{d_{\mathrm{y}}}} D_{KL}(\mathcal{N}_{\boldsymbol{Z}} \mid\mid \mathcal{N}_{\boldsymbol{Y} \cup \boldsymbol{y}}),$$

where $\mathcal{N}_{\boldsymbol{Z}}$ and $\mathcal{N}_{\boldsymbol{Y} \cup \boldsymbol{y}}$ are defined in Eq. (18) and Eq. (20) respectively.

As a result, inference in TGP is given as the solution to the following optimization problem (Bo and Sminchisescu, 2010):

$$\begin{aligned} \widehat{\boldsymbol{y}} = \operatorname*{argmin}_{\boldsymbol{y} \in \mathbb{R}^{d_{\mathrm{y}}}} \Big[& L(\boldsymbol{y}, \boldsymbol{y}) - 2\boldsymbol{l}(\boldsymbol{y})^\top \boldsymbol{u} \\ & - \eta \log \left[L(\boldsymbol{y}, \boldsymbol{y}) - \boldsymbol{l}(\boldsymbol{y})^\top \boldsymbol{L}^{-1} \boldsymbol{l}(\boldsymbol{y}) \right] \Big], \end{aligned} \tag{21}$$

where $\boldsymbol{u} = \boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{z})$, $\eta = K(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{k}(\boldsymbol{z})^\top \boldsymbol{u}$, $K(\boldsymbol{z}_i, \boldsymbol{z}_j) = \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{2\rho_{\mathrm{z}}^2}\right) + \lambda_{\mathrm{z}}\delta_{ij}$ and $L(\boldsymbol{y}_i, \boldsymbol{y}_j) = \exp\left(-\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2}{2\rho_{\mathrm{y}}^2}\right) + \lambda_{\mathrm{y}}\delta_{ij}$ are the Gaussian kernel function for image feature vector $\boldsymbol{z}$ and pose feature vector $\boldsymbol{y}$, ρ_{z} and ρ_{y} are the kernel bandwidth, $\boldsymbol{l}(\boldsymbol{y}) = [L(\boldsymbol{y}, \boldsymbol{y}_1), \ldots, L(\boldsymbol{y}, \boldsymbol{y}_{n_{\mathrm{tr}}})]^\top$, $\boldsymbol{k}(\boldsymbol{z}) = [K(\boldsymbol{z}, \boldsymbol{z}_1), \ldots, K(\boldsymbol{z}, \boldsymbol{z}_{n_{\mathrm{tr}}})]^\top$, and λ_{y} and λ_{z} are regularization parameters to avoid overfitting. This nonlinear optimization problem can be solved using a second order Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton optimizer with cubic polynomial line search for optimal step size selection (Bo and Sminchisescu, 2010).

A.3 Importance Weighting in TGP

Under covariate shift, the likelihood of Gaussian Process can be given as (Shimodaira, 2000)

$$\begin{aligned}&\prod_{i=1}^{n_{\mathrm{tr}}} p(y_i^{tr}|\boldsymbol{z}_i^{\mathrm{tr}}, \boldsymbol{b})^{w_\alpha(\boldsymbol{z}_i^{\mathrm{tr}})}\\ &\propto \prod_{i=1}^{n_{\mathrm{tr}}} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|w_\alpha^{\frac{1}{2}}(\boldsymbol{z}_i^{\mathrm{tr}})y_i^{tr} - w_\alpha^{\frac{1}{2}}(\boldsymbol{z}_i^{\mathrm{tr}})\boldsymbol{\phi}(\boldsymbol{z}_i^{\mathrm{tr}})^\top \boldsymbol{b}\|^2}{2\sigma^2}\right),\end{aligned} \tag{22}$$

where $w_\alpha(\boldsymbol{z})$ is the relative importance weight function.

Thus, the GP regression model under covariate shift can be represented by

$$w_\alpha^{\frac{1}{2}}(\boldsymbol{z})y_k = w_\alpha^{\frac{1}{2}}(\boldsymbol{z})\boldsymbol{\phi}(\boldsymbol{z})^\top \boldsymbol{b}_k + e_k,\ e_k \sim \mathcal{N}(0, \sigma^2). \tag{23}$$

That is, to achieve covariate shift adaptation in TGP, we need to simply re-weight each input and output by $w_\alpha^{\frac{1}{2}}(\boldsymbol{z})$. The optimization problem of IWTGP is therefore given by (Yamada *et al.*, 2012)

$$\begin{aligned}\widehat{\boldsymbol{y}} = \operatorname*{argmin}_{\boldsymbol{y}\in\mathbb{R}^{d_{\mathrm{y}}}} \Big[&L(\boldsymbol{y}, \boldsymbol{y}) - 2\boldsymbol{l}(\boldsymbol{y})^\top \boldsymbol{u}_w\\ &-\eta_w \log\Big[L(\boldsymbol{y}, \boldsymbol{y}) - \boldsymbol{l}(\boldsymbol{y})^\top \boldsymbol{L}_{\boldsymbol{w}}^{-1}\boldsymbol{l}(\boldsymbol{y})\Big]\Big],\end{aligned} \tag{24}$$

where $K(\boldsymbol{z}_i^{\mathrm{tr}}, \boldsymbol{z}_j^{\mathrm{tr}}) = \exp\left(-\frac{\|\boldsymbol{z}_i^{\mathrm{tr}} - \boldsymbol{z}_j^{\mathrm{tr}}\|^2}{2\rho_{\mathrm{z}}^2}\right) + \lambda_{\mathrm{z}}\widetilde{\delta}_{ij}$ and $L(\boldsymbol{y}_i^{\mathrm{tr}}, \boldsymbol{y}_j^{\mathrm{tr}}) = \exp\left(-\frac{\|\boldsymbol{y}_i^{\mathrm{tr}} - \boldsymbol{y}_j^{\mathrm{tr}}\|^2}{2\rho_{\mathrm{y}}^2}\right) + \lambda_{\mathrm{y}}\widetilde{\delta}_{ij}$, $\widetilde{\delta}_{ij}$ takes $w(\boldsymbol{z}_i^{\mathrm{tr}})^{-1}$ if $i = j$ and zero if $i \neq j$, $\boldsymbol{u}_w = \boldsymbol{K}_{\boldsymbol{w}}^{-1}\boldsymbol{k}(\boldsymbol{z}^{\mathrm{tr}})$, $\eta_w = K(\boldsymbol{z}^{\mathrm{tr}}, \boldsymbol{z}^{\mathrm{tr}}) - \boldsymbol{k}(\boldsymbol{z})^\top \boldsymbol{u}_w$, and $w(\boldsymbol{z}^{\mathrm{tr}}) = \frac{p_{\mathrm{te}}(\boldsymbol{z}^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{z}^{\mathrm{tr}})}$ is called the *importance weight* function, which is used for compensating sample selection bias. The importance weight can be directly estimated from training and test image data sets $\{\boldsymbol{z}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\boldsymbol{z}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ by direct importance weight estimation such as relative unconstrained least-squares importance fitting (RuLSIF) (Yamada *et al.*, 2013).