A Supplementary Material for *A Geometric take on Metric Learning*

The supplementary material contains proofs for the theorems found in the paper as well as further results that were left out due to page constraints. When reading the proofs it can be convenient to remember the dimensionality of the individual parts of the equations, which we briefly summarize here:

$$\begin{array}{lll} \mathbf{c}(\lambda) \in \mathbb{R}^{D\times 1} & \mathbf{c}'(\lambda) \in \mathbb{R}^{D\times 1} & \mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda) \in \mathbb{R}^{D^2\times 1} \\ \mathbf{M}(\mathbf{c}(\lambda)) \in \mathbb{R}^{D\times D} & \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right] \in \mathbb{R}^{D^2\times 1} & \frac{\partial \mathsf{vec}[\mathbf{M}(\mathbf{c}(\lambda))]}{\partial \mathbf{c}(\lambda)} \in \mathbb{R}^{D^2\times D} \end{array}$$

A.1 Proof of Theorem 2

We remind the reader that we need to minimize *curve length* (eq. 4). Conveniently, these minima coincides with those of the *curve energy* [11] defined as

$$\text{Energy}(\mathbf{c}) = \int_0^1 L\mathrm{d}\lambda = \int_0^1 \mathbf{c}'(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda))\mathbf{c}'(\lambda)\mathrm{d}\lambda \ . \tag{14}$$

At times, we will find it useful to express L in terms of Kronecker products:

$$L = \mathbf{c}'(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda))\mathbf{c}'(\lambda) = \left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right] \ , \tag{15}$$

where $\otimes$ denotes the Kronecker product and $\mathsf{vec}\left[\cdot\right]$ unfolds a matrix to a vector by stacking its columns; see the book by Magnus and Neudecker [13] for details on this notation.

We compute the derivative of this expression using the Euler-Lagrange equation

$$\frac{\partial L}{\partial \mathbf{c}} = \frac{\mathrm{d}}{\mathrm{d}\lambda}\frac{\partial L}{\partial \mathbf{c}'} \ . \tag{16}$$

We derive the individual terms of this equation below.

$$\frac{\partial L}{\partial \mathbf{c}(\lambda)} = \frac{\partial}{\partial \mathbf{c}(\lambda)}\left[\left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]\right] \tag{17}$$

$$= \left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right] \ . \tag{18}$$

$$\frac{\partial L}{\partial \mathbf{c}'(\lambda)} = 2\mathbf{c}'(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda)) \tag{19}$$

$$= 2\mathsf{vec}\left[\mathbf{c}'(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda))\right] \tag{20}$$

$$= 2\left(\mathbf{I}_D\otimes\mathbf{c}'(\lambda)^T\right)\mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right] \ . \tag{21}$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\frac{\partial L}{\partial \mathbf{c}'(\lambda)} = 2\frac{\mathrm{d}\left(\mathbf{I}_D\otimes\mathbf{c}'(\lambda)^T\right)}{\mathrm{d}\lambda}\mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right] + 2\left(\mathbf{I}_D\otimes\mathbf{c}'(\lambda)^T\right)\frac{\mathrm{d}\mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\mathrm{d}\lambda} \tag{22}$$

$$= 2\left(\mathbf{I}_D\otimes\mathbf{c}''(\lambda)^T\right)\mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right] + 2\left(\mathbf{I}_D\otimes\mathbf{c}'(\lambda)^T\right)\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right]\mathbf{c}'(\lambda) \tag{23}$$

$$= 2\mathbf{c}''(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda)) + 2\left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right] \ . \tag{24}$$

Here $\mathbf{I}_D$ denotes the $D\times D$ identity matrix.

All this can be combined to give

$$\begin{aligned} \left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right] &= 2\mathbf{c}''(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda)) \\ &\quad + 2\left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right] \ \Leftrightarrow \end{aligned} \tag{25}$$

$$-\frac{1}{2}\left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right)^T\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right] = \mathbf{c}''(\lambda)^T\mathbf{M}(\mathbf{c}(\lambda)) \ \Leftrightarrow \tag{26}$$

$$\mathbf{M}(\mathbf{c}(\lambda))\mathbf{c}''(\lambda) = -\frac{1}{2}\left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right]^T\left(\mathbf{c}'(\lambda)\otimes\mathbf{c}'(\lambda)\right) \ , \tag{27}$$

which concludes the proof.

A.2 Proof of Theorem 3

First we consider the following general weighting scheme

$$w_r = \frac{\tilde{w}_r}{\sum_{j=1}^{R} \tilde{w}_j} \quad . \tag{28}$$

We will consider different choices of $\tilde{w}_r$, but first we compute the derivative of the metric tensor wrt. $\mathbf{c}$.

$$\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c})\right]}{\partial \mathbf{c}} = \sum_{r=1}^{R} \mathsf{vec}\left[\mathbf{M}_r\right] \frac{\partial w_r}{\partial \mathbf{c}} \tag{29}$$

$$\frac{\partial w_r}{\partial \mathbf{c}} = \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \left(\frac{\partial \tilde{w}_r}{\partial \mathbf{c}} \sum_{j=1}^{R} \tilde{w}_j - \tilde{w}_r \sum_{j=1}^{R} \frac{\partial \tilde{w}_j}{\partial \mathbf{c}}\right) \quad . \tag{30}$$

In the following we derive $\frac{\partial \mathsf{vec}[\mathbf{M}(\mathbf{c})]}{\partial \mathbf{c}}$ for two squared exponential schemes. Other schemes can easily be derived as long as the chosen weights are smooth.

A.2.1 Global Weights: $\tilde{w}_r = \exp\left(-\frac{1}{2}\|\mathbf{c}(\lambda) - \mathbf{x}_r\|_{\Gamma}^2\right)$

We first consider a weighting scheme using a global metric

$$\tilde{w}_r(\mathbf{c}) = \exp\left(-\frac{1}{2}\|\mathbf{c} - \mathbf{x}_r\|_{\Gamma}^2\right) \quad , \tag{31}$$

where Γ is a metric tensor. The derivative is given as

$$\frac{\partial \tilde{w}_r}{\partial \mathbf{c}} = -\tilde{w}_r \left(\mathbf{c} - \mathbf{x}_r\right)^T \Gamma \tag{32}$$

$$\frac{\partial w_r}{\partial \mathbf{c}} = -\tilde{w}_r \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \left(\left(\mathbf{c} - \mathbf{x}_r\right)^T \Gamma \sum_{j=1}^{R} \tilde{w}_j - \sum_{j=1}^{R} \tilde{w}_j \left(\mathbf{c} - \mathbf{x}_j\right)^T \Gamma\right) \tag{33}$$

$$= -\tilde{w}_r \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \sum_{j=1}^{R} \tilde{w}_j \left(\left(\mathbf{c} - \mathbf{x}_r\right)^T \Gamma - \left(\mathbf{c} - \mathbf{x}_j\right)^T \Gamma\right) \tag{34}$$

$$= -\tilde{w}_r \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \sum_{j=1}^{R} \tilde{w}_j \left(\left(\mathbf{x}_j - \mathbf{x}_r\right)^T \Gamma\right) \quad . \tag{35}$$

This gives the following derivative

$$\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c})\right]}{\partial \mathbf{c}} = -\left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \sum_{r=1}^{R} \tilde{w}_r \mathsf{vec}\left[\mathbf{M}_r\right] \sum_{j=1}^{R} \tilde{w}_j \left(\left(\mathbf{x}_j - \mathbf{x}_r\right)^T \Gamma\right) \quad . \tag{36}$$

A.2.2 Local Weights: $\tilde{w}_r = \exp\left(-\frac{\rho}{2}\|\mathbf{c}(\lambda) - \mathbf{x}_r\|_{\mathbf{M}_r}^2\right)$

We now consider using the prototype metric tensors to determine the weights:

$$\tilde{w}_r(\mathbf{c}) = \exp\left(-\frac{\rho}{2}\|\mathbf{c} - \mathbf{x}_r\|_{\mathbf{M}_r}^2\right) \quad . \tag{37}$$

We can then compute the derivative as

$$\frac{\partial \tilde{w}_r}{\partial \mathbf{c}} = -\rho \tilde{w}_r \left(\mathbf{c} - \mathbf{x}_r\right)^T \mathbf{M}_r \tag{38}$$

$$\frac{\partial w_r}{\partial \mathbf{c}} = -\rho \tilde{w}_r \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \left(\left(\mathbf{c} - \mathbf{x}_r\right)^T \mathbf{M}_r \sum_{j=1}^{R} \tilde{w}_j - \sum_{j=1}^{R} \tilde{w}_j \left(\mathbf{c} - \mathbf{x}_j\right)^T \mathbf{M}_j\right) \tag{39}$$

$$= -\rho \tilde{w}_r \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \sum_{j=1}^{R} \tilde{w}_j \left(\left(\mathbf{c} - \mathbf{x}_r\right)^T \mathbf{M}_r - \left(\mathbf{c} - \mathbf{x}_j\right)^T \mathbf{M}_j\right) \quad . \tag{40}$$

This gives the following derivative

$$\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c})\right]}{\partial \mathbf{c}} = -\rho \left(\sum_{j=1}^{R} \tilde{w}_j\right)^{-2} \sum_{r=1}^{R} \tilde{w}_r \mathsf{vec}\left[\mathbf{M}_r\right] \sum_{j=1}^{R} \tilde{w}_j \left(\left(\mathbf{c} - \mathbf{x}_r\right)^T \mathbf{M}_r - \left(\mathbf{c} - \mathbf{x}_j\right)^T \mathbf{M}_j\right) \quad . \tag{41}$$

A.3 From 2^{nd} order to 1^{st} order

When computing geodesics we have to solve a system of 2^{nd} order ODE's, which we solve by rewriting it as a system of 1^{st} order ODE's. This is standard practice, but we briefly describe it here for the sake of completeness.

We are interested in solving a system of the form

$$\mathbf{c}''(\lambda) = \mathbf{f}(\lambda, \mathbf{c}, \mathbf{c}') = -\frac{1}{2}\mathbf{M}^{-1}(\mathbf{c}(\lambda)) \left[\frac{\partial \mathsf{vec}\left[\mathbf{M}(\mathbf{c}(\lambda))\right]}{\partial \mathbf{c}(\lambda)}\right]^T \left(\mathbf{c}'(\lambda) \otimes \mathbf{c}'(\lambda)\right) \ . \tag{42}$$

We now let

$$\mathbf{g}(\lambda) = \mathbf{c}'(\lambda) \ . \tag{43}$$

Now eq. 42 can be re-expressed by solving for both $\mathbf{c}$ and $\mathbf{c}'$ as

$$\begin{bmatrix} \mathbf{c}'(\lambda) \\ \mathbf{g}'(\lambda) \end{bmatrix} = \begin{bmatrix} \mathbf{g}(\lambda) \\ \mathbf{f}(\lambda, \mathbf{c}, \mathbf{g}) \end{bmatrix} \ , \tag{44}$$

which is a system of 1^{st} order equations.

A.4 Algorithmic Details

We use standard algorithms for doing statistics on manifolds. Specifically, we use the mean value estimator from Pennec [18] and the standard PGA model from Fletcher et al. [16]. For completeness, these are presented in Algorithm 1 and 2 respectively. The algorithm for regression is identical to the PGA algorithm, with the exception that the PCA part is replaced by standard linear regression.

The algorithms rely on functions for computing geodesics and exponential maps on the manifold. We compute these by setting up the boundary value problem in (10) and the initial value problem in (12). We solve these numerically using standard off-the-shelf solver; specifically we use `bvp4c` and `ode45` from Matlab.

input : Training data $\mathbf{p}_{1:N}$; and manifold structure in the form of metric tensors $\mathbf{M}_{1:R}$ and their positions $\mathbf{x}_{1:R}$
output: Mean value μ

```
// Compute initial mean:
```
$\mu_1 \leftarrow \frac{1}{N}\sum_{n=1}^{N} \mathbf{p}_n$;
```
// Iteratively improve mean value estimate:
```
for $i \leftarrow 2 \ldots$ **do**
```
    // Compute geodesics to the current mean estimate:
```
    parallel for $n \leftarrow 1$ **to** N **do**
        $\gamma_n \leftarrow$ `compute_geodesic(`$\mathbf{p}_n, \mu_{i-1}, \mathbf{M}_{1:R}, \mathbf{x}_{1:R}$`)`;
        $\mathbf{l}_n \leftarrow \frac{\gamma_n'(0)}{\|\gamma_n'(0)\|}\mathrm{Length}(\gamma_n)$;
    end
```
    // Compute tangent space mean:
```
    $\hat{\mu}_i \leftarrow \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}_n$;
```
    // Map tangent space mean to the feature space:
```
    $\mu_i \leftarrow$ `exponential_map(`$\mu_{i-1}, \mu_i, \mathbf{M}_{1:R}, \mathbf{x}_{1:R}$`)`;
end

Algorithm 1: Computing Karcher means.

input : Training data $\mathbf{p}_{1:N}$; and manifold structure in the form of metric tensors $\mathbf{M}_{1:R}$ and their positions $\mathbf{x}_{1:R}$
output: Principal geodesics $\gamma_{\mathbf{v}_{1:D}}$

`// Compute mean value on manifold:`
$\mu \leftarrow$ `karcher_mean` $(\mathbf{p}_{1:N}, \mathbf{M}_{1:R}, \mathbf{x}_{1:R})$;

`// Compute geodesics to the mean:`
parallel for $n \leftarrow 1$ **to** N **do**
  $\gamma_n \leftarrow$ `compute_geodesic` $(\mathbf{p}_n, \mu, \mathbf{M}_{1:R}, \mathbf{x}_{1:R})$;
  $\mathbf{l}_n \leftarrow \frac{\gamma_n'(0)}{\|\gamma_n'(0)\|}\text{Length}(\gamma_n)$;
end

`// Perform PCA in the tangent space:`
$\mathbf{v}_{1:D} \leftarrow$ `PCA` $(\mathbf{l}_{1:N})$;

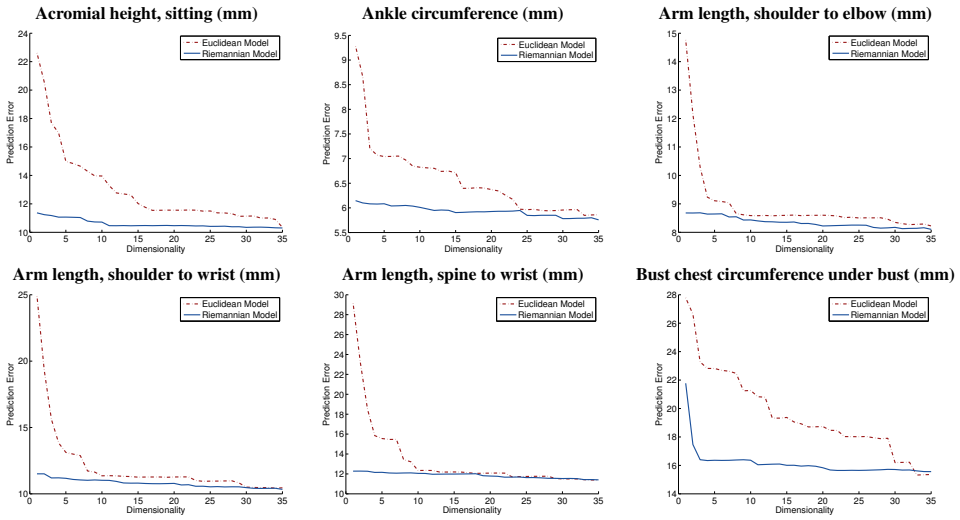
`// Map results back to the feature space:`
for $d \leftarrow 1$ **to** D **do**
  $\gamma_{\mathbf{v}_d} \leftarrow$ `exponential_map` $(\mu, \mathbf{v}_d, \mathbf{M}_{1:R}, \mathbf{x}_{1:R})$;
end

Algorithm 2: Principal Geodesic Analysis (PGA).

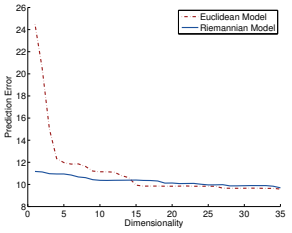
A.5 Further Results on Human Body Shapes

This section contains plots for the regression errors, when predicting body measurements from body shapes. The experiments follow those in Sec. 4.1 of the paper, i.e.

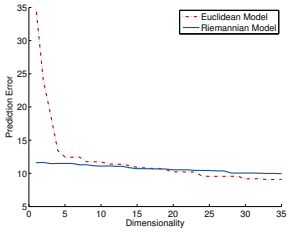
1. First, we whiten the data to ensure that all changes in variance is due to the change of the metric.
2. We then sort the shape data according to the specific measurement (e.g. *arm length*) and split it in 5 equal-sized clusters.
3. For each cluster we learn a LMNN metric [5], which pushes the different clusters apart. This will locally introduce variance in the directions that pushes the clusters apart. Globally, the learned metrics stretches the feature space in the directions that are most important for the specific measurement.
4. We then construct a Riemannian metric according to (6) and compute the mean of the data according to this metric (Algorithm 1).
5. We then compute geodesics between each data point and the mean, and map the data into the tangent space at the mean using the logartihmic map (11).
6. In this Euclidean representation of the data, we perform linear regression to predict the measurement. The measurement of an unseen point is predicted by mapping it into the tangent space and applying the Euclidean model.



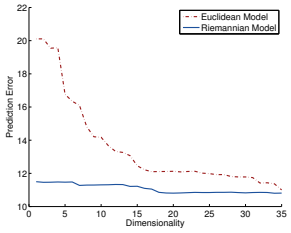
Buttock knee length (mm)



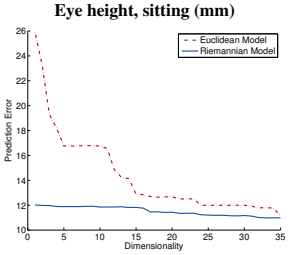
Crotch height (mm)



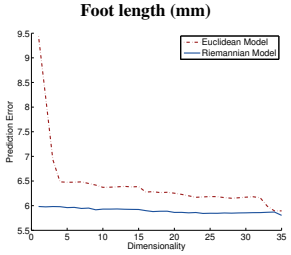
Elbow height, sitting (mm)



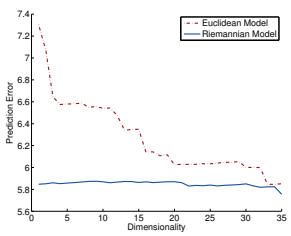
Eye height, sitting (mm)



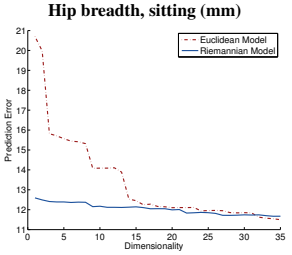
Foot length (mm)



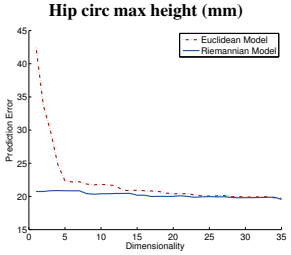
Hand circumference (mm)



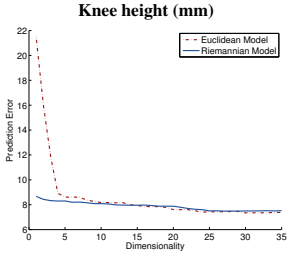
Hip breadth, sitting (mm)



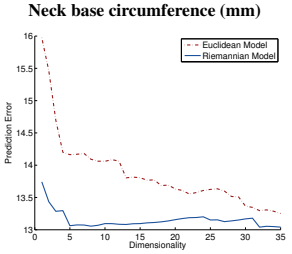
Hip circ max height (mm)



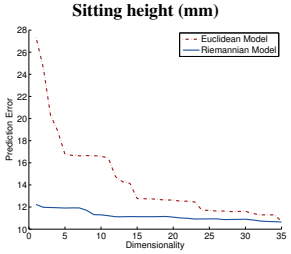
Knee height (mm)



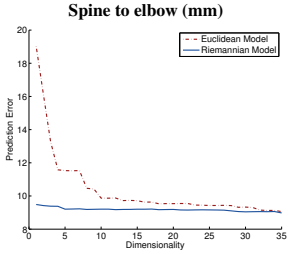
Neck base circumference (mm)



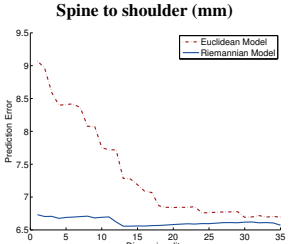
Sitting height (mm)



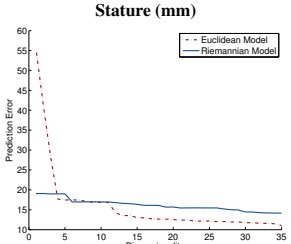
Spine to elbow (mm)



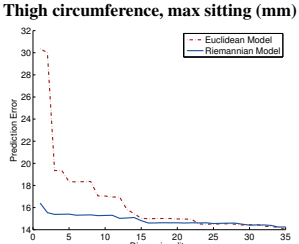
Spine to shoulder (mm)



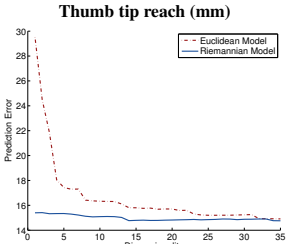
Stature (mm)



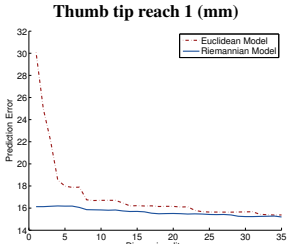
Thigh circumference, max sitting (mm)



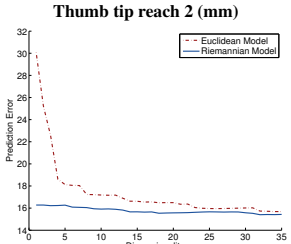
Thumb tip reach (mm)

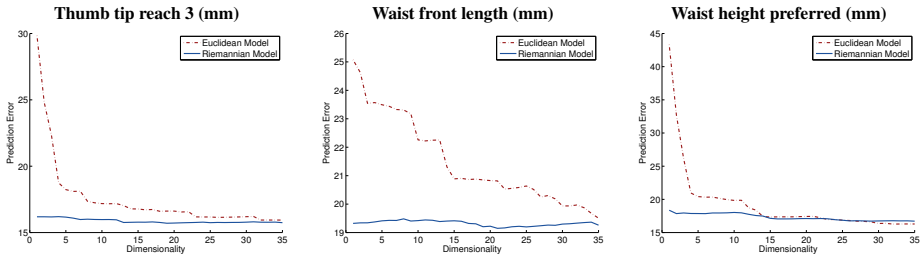


Thumb tip reach 1 (mm)



Thumb tip reach 2 (mm)





A.6 MNIST Dimensionality Reduction Plots

In Sec. 4.2 we considered PGA on the MNIST data set, where we learned one metric per class using LMNN [5], which we associated with the class center. Fig. 6 show the first two principal components of the digits 4, 7 and 9, computed according to the Euclidean metric and in the tangent space of the manifold implied by the local metrics. As can be seen, the two representations are quite different, but it is not clear that one is better than the other. To quantify the behavior we also perform a classification study.

We perform nearest neightbor classification of 60,000 data points from all ten classes, and test on 1,000 separate data points. We learn one LMNN metric per class, which we associate with the mean of the class. From this we construct a Riemannian manifold from (6), compute the mean value on the manifold, map the data to the tangent space at the mean and perform ordinary PCA in the tangent space, i.e. PGA. To study the effect of dimensionality reduction according to the learned metrics, we measure the classification error while we gradually reduce the dimensionality of the data using both the learned metrics and, for comparison, the ordinary Euclidean metric. The results are plotted in Fig. 6. As can be seen, the model using the learned metrics slightly out-performs the baseline Euclidean metric. The interesting result is that this result holds as the dimensionality is reduced, which shows that the Riemannian PGA captures the important parts of the learned metric.

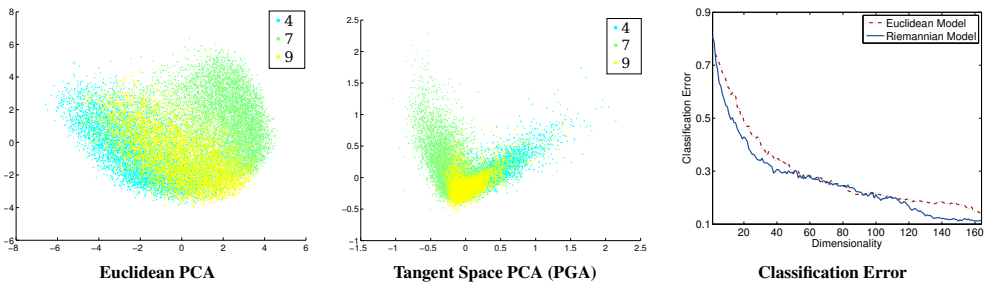


Figure 6: *Left:* The MNIST data expressed in first two principal components computed according to the Euclidean metric. *Center:* The same data expressed in the first two principal components computed in the tangent space of the manifold implied by the local LMNN metrics. *Right:* The classification error as a function of the dimensionality of the Euclidean and the Riemannian models.