FootNet: An Efficient Convolutional Network for Multiview 3D Foot Reconstruction

Felix Kok, James Charles, and Roberto Cipolla

Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK
{cyk28,jjc75,rc10001}@cam.ac.uk

Abstract. Automatic biometric analysis of the human body is normally reserved for expensive customisation of clothing items e.g. for sports or medical purposes. These systems are usually built upon photogrammetric techniques currently requiring a rig and well calibrated cameras. Here we propose building on advancements in deep learning as well as utilising technology present in mobile phones for cheaply and accurately determining biometric data of the foot. The system is designed to run efficiently in a mobile phone app where it can be used in uncalibrated environments and without rigs. By scanning the foot with the phone camera, our system recovers both the 3D shape as well as the scale of the foot, opening the door way for automatic shoe size suggestion. Our contributions are (1) an efficient multiview feed forward neural network capable of inferring foot shape and scale, (2) a system for training from completely synthetic data and (3) a dataset of multiview feet images for evaluation. We fully ablate our system and show our design choices to improve performance at every stage. Our final design has a vertex error of only 1mm (for 25cm long synthetic feet) and 4mm error in foot length on real feet.

1 Introduction

Footwear is an essential clothing item for all age groups and genders, serving many practical purposes, such as protection, but also typically worn as a fashion item. It is conventional and often vital for one to physically try on a pair of ready-to-wear shoes prior to deciding upon a purchase. This is cumbersome for in-store shopping but very inefficient and environmentally damaging for online shopping. In this setting, it is standard for many shoes to be transported back and forth between warehouse and customer to accommodate for this try-on process.

Foot length-to-size charts can be easily found on the internet but the conversion tends to vary from brand to brand. Length alone is also often not sufficient to characterise the entire shape of the foot and other measurements such as foot width and instep girth are important for correct fitting. Therefore, a method for easily obtaining a 3D model of the foot would be beneficial as it allows customers to virtually try on shoes to find the best size and shape.

Many products for 2D/3D foot scanning already exist on the market such as those developed by VoxelCare [1] and Vorum [2] but these devices tend to be expensive and are usually not targeted at common shoe customers.

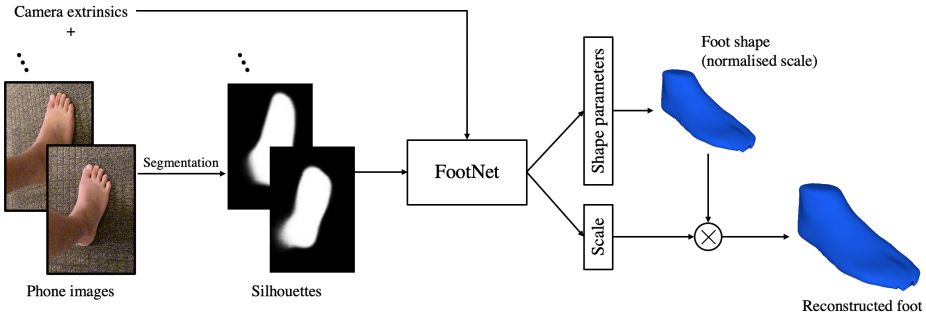


Fig. 1: Overview of the foot reconstruction framework

In this paper, we propose a novel end-to-end framework, which we call FootNet, that reconstructs a scale accurate 3D mesh representation of the foot from multiple 2D images. Reconstruction is very quick and can be computed directly on a smart phone.

2 Related work

Traditional methods take a geometrical approach to tackle the reconstruction problem. Examples of commonly used techniques are passive/active triangulation [1, 2, 3, 4, 5, 6] and space carving [7]. Deep learning has been tremendously successful in tackling vision related problems including 3D reconstruction. CNNs for object reconstruction typically have an encoder that maps the input into a latent variable or feature vector, which is then decoded into the desired output depending on how the 3D shape is represented. Common representations include voxel grids, meshes and point clouds.

Voxel grids. A standard approach is to use up-convolutional layers to directly regress the 3D voxel occupancy probabilities from the latent variables [8, 9, 10]. With the availability of inexpensive 2.5D depth sensors (e.g. Microsoft Kinect), methods were proposed to reconstruct objects from depth maps [11, 12, 13]. For example, MarrNet [12] uses a two-stage network, where the first stage is an encoder-decoder architecture that predicts the 2.5D sketches (depth, surface normals, and silhouette), and the second stage is another encoder-decoder network that outputs a voxelised shape. These methods have generally been successful in reconstruction tasks but the output resolution is limited due to the high memory requirements, as the memory scales cubically with the voxel resolution. The methods mentioned above produce grids of resolution 32^3 to 64^3, except for MarrNet whose output resolution is 128^3. Approaches such as space partitioning [14, 15] and coarse-to-fine refinement [16, 17, 18] were able to increase the output voxel size to 256^3 or 512^3.

Meshes. Meshes are less demanding in memory but they are not regularly structured so the network architectures have to be specifically designed. One approach is to start with a template such as a sphere and deform the template

to output the predicted 3D shape [19, 20]. Kato *et al.* [19] proposed a method to approximate the gradient for rendering which enables integration of rendering a mesh into a neural network. Wang *et al.* [20] represents the 3D mesh in a graph-based CNN and predicts the shape by progressively deforming an ellipsoid.

Multiview networks. The problem of self-occlusion could be overcome by providing more than one viewpoint, especially in reconstructing novel shapes. Several methods were proposed to combine the information from different viewpoints. For example, silhouettes from multiple views can be combined at the input as separate channels and then passed through convolutional layers, or they can be passed into separate convolutional blocks and the outputs are concatenated [21]. The number of viewpoints would be fixed for such models. Choy *et al.* [22] uses a LSTM framework to combine a variable number of views but the output is not consistent if the order of input views is altered. Wiles and Zisserman's [23] uses max-pooling to combine the encoded latent feature vectors from multiple views so that the result is not affected by the order of input images and it could generalise to any number of views.

Foot reconstruction. Several methods were developed specifically for foot shape reconstruction. For example, Amstutz *et al.* [24] reduces the 3D vertices of feet to only 12 parameters while preserving 92% of the shape variation, using PCA decomposition on a foot dataset. Given multiple images of a foot from different viewpoints, reconstruction is done by optimising the pose parameters, shape parameters and scale. However, their system operates in a very constrained setting, using a camera rig, structured lighting and physical aids for background subtraction. Our solution is for in the wild use, using a mobile phone. Another approach [25] uses deep learning to infer the foot shape from a single depth map by synthesising a new view that contains information of the foot missing from the input. Unfortunately, this method requires a depth sensor such as the Microsoft Kinect to operate.

3 Overview

Our system is illustrated in Fig 1 and is broken down into three parts:

Acquisition. Our system takes multiple photos of the target from various viewpoints surrounding the foot. Using a smart phone camera we utilise the AR features (ARKit/ARCore) and attach to each image the real world camera extrinsics. The RGB images are preprocessed by passing them through a foot segmentation network.

3D inference using FootNet. A deep network ingests the silhouette and camera pose data to infer foot length as well as shape. This regression network (FootNet) takes inspiration from the architecture of SilNet [23] and is able to handle any number of input viewpoints without being affected by the order of inputs. Compared to SilNet which was shown to only handle 1 degree of freedom in camera pose, FootNet is built to handle all 6. In addition, FootNet regresses to

a dense mesh reconstruction rather than a voxel grid and incorporates an efficient encoder based on MobileNet [26] to allow mobile implementation. We also show FootNet works on real data whereas SilNet was only tested on synthetic data.

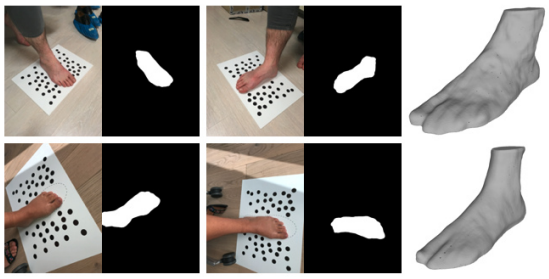
Foot shape and scale. Foot shape is paramaterised by a PCA foot model trained from 3D scans of people's feet using a multi-view stereo (MVS) system [27]. We train our deep network to infer these parameters from synthetic data only. Scale of the foot is inferred as a separate output. We next describe our method in detail.

4 Methods

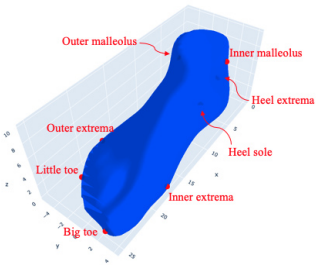
Our network is trained on synthetic foot silhouettes generated using arbitrary foot shapes and camera poses. A PCA based 3D foot mesh is used here and silhouettes are rendered by artificially adjusting camera extrinsics and sampling shape from the PCA model.

4.1 3D foot mesh parameterisation

3D meshes of over 1600 feet are obtained using a MVS system[1] [27]. We apply PCA to this foot dataset, similar to Amstutz *et al.* [24], expressing changes in foot shape based on 10 PCA parameters. The foot mesh is composed of 1602 vertices. Fig 2(a) illustrates the data collection pipeline showing calibration pattern used for multi-view stereo and in Fig 2(b) the PCA based foot mesh with annotated vertex points representing various anatomical positions on the foot. A comparison of example dense foot meshes and their compact PCA representation is shown in Fig 3.



(a) RGB images, segmentation masks and dense meshes



(b) Labelled keypoints

Fig. 2: Constructing the PCA based model. Data collection using MVS is shown in (a) and the final PCA based model with vertex annotations is shown in (b)

From Fig 4, we see that the first coefficient corresponds to the roundness of the toes, the second corresponding to width and thickness, the fourth corresponding to height of the big toe.

[1] `https:/snapfeet.io/en`

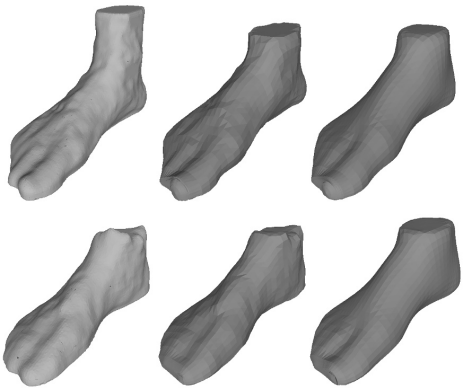


Fig. 3: Comparison of the dense meshes (left column), meshes reconstructed using 1602 PCA coefficients (middle column) and meshes reconstructed using 10 PCA coefficients (right column).

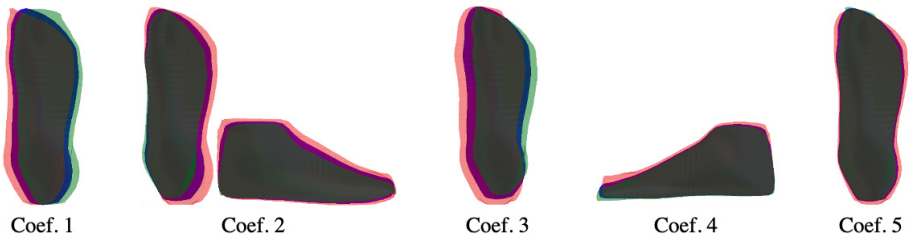


Fig. 4: Change in foot shape when one of the PCA coefficients is varied while the others are fixed. Blue: mean foot. Red: coefficient set to largest in dataset. Green: coefficient set to smallest in dataset.

4.2 Foot mesh reconstruction model (FootNet)

Having compressed the foot shape to 10 PCA coefficients, our goal is to construct and train a regression model that predicts the coefficients and size given foot silhouettes and the corresponding camera poses. We propose a multi-view framework, inspired by Wiles and Zisserman [23], that can handle any number of viewpoints and is not affected by the order of the inputs. The overall framework is displayed in Fig 5. To allow more flexibility in the shape prediction, we add two extra output units representing the width deformation, k_w, and height deformation, k_h, which scale the foot vertices in the horizontal (inside to outside) and vertical (sole to ankle) directions respectively:

$$\mathbf{V}_{original} = \begin{bmatrix} \mathbf{v}_x & \mathbf{v}_y & \mathbf{v}_z \end{bmatrix} \quad \longrightarrow \quad \mathbf{V}_{deformed} = \begin{bmatrix} \mathbf{v}_x & k_w\mathbf{v}_y & k_h\mathbf{v}_z \end{bmatrix} \tag{1}$$

At test time, foot segmentation is produced by a separate CNN based on ENet [28] trained on the 1601 collected foot images and silhouettes obtained from reprojection of the dense foot meshes. The main focus of this paper is on inferring the shape and scale of the foot and we are agnostic to the method used for segmentation.

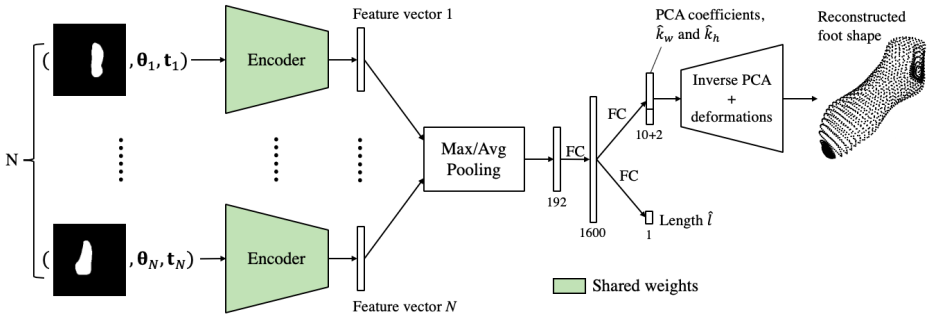


Fig. 5: Foot mesh reconstruction model architecture (FootNet).

Loss function. In our framework, we consider shape prediction and length prediction separate tasks each having its own loss function.

Since the ground truth foot shapes are available, an intuitive loss for shape is the 3D vertex error, i.e., mean vertex error on the 1602-vertex point cloud. To compute the shape loss, L_{shape}, the predicted and ground truth shapes are both scaled to 25 cm long and the corresponding vertices ($\mathbf{V}$ and $\hat{\mathbf{V}}$) are compared:

$$L_{vertex}(\mathbf{V}, \hat{\mathbf{V}}) = \frac{1}{m} \sum_{i=1}^{m} d(\mathbf{v}_i, \hat{\mathbf{v}}_i) \tag{2}$$

where $\mathbf{v}_i$ and $\hat{\mathbf{v}}_i$ are the i-th row of $\mathbf{V}$ and $\hat{\mathbf{V}}$ (i.e. 3D coordinates of a vertex) respectively and $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$.

To improve the robustness, the Huber loss [29] is applied to each vertex and the mean is taken.

$$L_{huber}(a; \delta) = \begin{cases} \frac{1}{2}a^2, & |a| \leq \delta, \\ \delta\left(|a| - \frac{1}{2}\delta\right), & otherwise \end{cases} \tag{3}$$

$$L_{shape}(\mathbf{V}, \hat{\mathbf{V}}; \delta) = \frac{1}{m} \sum_{i=1}^{m} L_{Huber}\left(d(\mathbf{v}_i, \hat{\mathbf{v}}_i); \delta_{shape}\right) \tag{4}$$

The scale loss, L_{scale}, is the Huber loss on the foot length and the overall loss function used to train the network is the *weighted loss* between L_{shape} and L_{scale}:

$$L_{scale} = L_{Huber}(l - \hat{l}; \delta_{scale}) \tag{5}$$

$$L = w_{shape} L_{shape} + w_{scale} L_{scale} \tag{6}$$

Architecture. The model has a 3-stage architecture (Fig 5): *encoding*, *combining* and *decoding*. The encoder is given a silhouette and camera pose ($\boldsymbol{\theta}$ and $\mathbf{t}$) as inputs and it computes a 1D feature vector. The encoder can be replicated as many times as there are number of views. Since the parameters of all encoders are shared, memory is saved by running the encoder sequentially over input views. The N feature vectors are combined into a single feature vector by a pooling

layer. Finally, the decoder regresses the PCA coefficients, the two deformation scaling factors (k_w and k_h) and foot length through two fully connected layers. Linear activation functions are applied to all 13 output units. The reconstructed shape is obtained by applying the inverse PCA to the coefficients, scaling the vertices horizontally and vertically by k_w and k_h respectively (Eq. 1), and finally scaling the overall foot according to the predicted length. A major advantage of this framework is its ability to take into account any number of images and the prediction is not affected by the order of input views since the features are combined by pooling operations.

The encoder (Fig 6) consists of two branches: the image branch and camera branch. The silhouette is passed through the image branch which is a CNN based on MobileNet. The top fully connected layer and softmax layer of the MobileNet are removed so that it outputs a $7\times7\times1024$ tensor. The camera branch computes the sin and cos of the three camera angles ($\boldsymbol{\theta}$), combines them with the camera's position vector ($\mathbf{t}$) and passes them through two fully connected layers. The output of the camera branch is broadcast and concatenated to the image branch. Two further convolutional layers are applied to encode the combined output of the two branches to a single 1D feature vector. Dropout at rate 0.5 is applied to the two fully connected layers in the camera branch as well as the output from each encoder before they are combined in the pooling layer. These layers are added to help mitigate noise in camera pose and encourage the network to do well on all views. By incorporating the camera pose, we want the model to learn which specific views of the foot are responsible for specific parts or features of the foot.

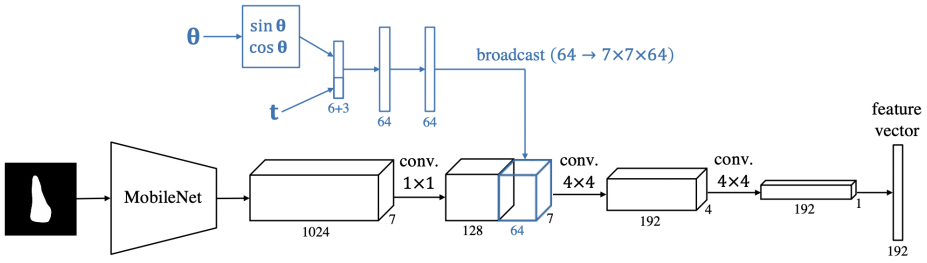


Fig. 6: Encoder architecture.

5 Data

5.1 Synthetic silhouettes

We generate synthetic foot silhouettes to train the model as it enables us to project the foot with any camera pose desired. We also aim to cover the "foot shape space" more thoroughly by randomly sampling foot shapes in the space, rather than being restricted to those scanned using MVS. A foot silhouette is generated by randomly sampling model parameters, scale and camera pose. The model is then rendered using a perspective projection.

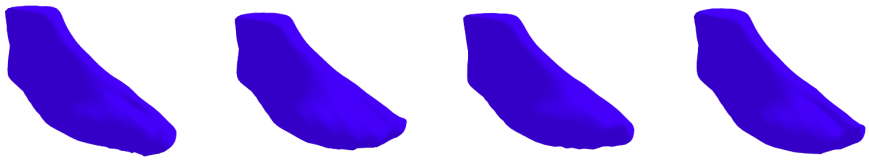


Fig. 7: Examples of randomly sampled foot shapes.

Examples of these randomly generated foot shapes are displayed in Fig 7. To project the sampled foot mesh, we use a pipeline illustrated in Fig 8. The camera pose is sampled such that it points approximately at the foot centre. The ranges from which the parameters for silhouette generation are uniformly sampled from are shown in Table 1. For each foot, 7 silhouettes are generated and we ensure that α is roughly uniformly distributed across the 7 views so that different sides of the foot are covered in a set of silhouettes. 12500 silhouette sets are generated and split 75/10/15 into train/val/test. Sample sets are shown in Fig 9.

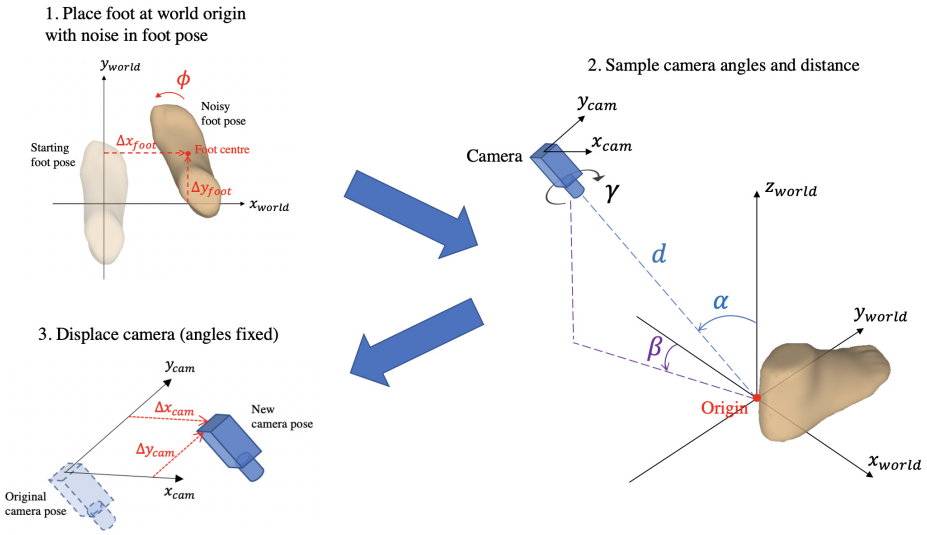


Fig. 8: Pipeline for sampling a foot pose and camera pose. A silhouette is produced by perspective projection of the foot mesh.

5.2 Real foot image datasets

We collected 2 datasets of real foot images along with camera extrinsics recovered from the smart phone. The 3-view dataset consists of 22 sets of foot images, each containing 3 views (inside, top, outside) of the foot from 11 people. The 20-view dataset consists of 5 sets of foot images, each containing 20 views. For each set of images, the foot is fixed in the same position and the view angles are roughly

Table 1: Ranges of parameters for sampling a foot pose and camera pose using the pipeline in Fig 8.

Parameter	Range
Foot length	19 to 32 cm
k_w, k_h	0.93 to 1.07
Δx_{foot}, Δy_{foot}	-3 to $+3$ cm
ϕ	-5° to $+5^\circ$
d	45 to 75 cm
α	-70° to $+70^\circ$
β, γ	-10° to $+10^\circ$
Δx_{cam}	-5 to $+5$ cm
Δy_{cam}	-10 to $+10$ cm

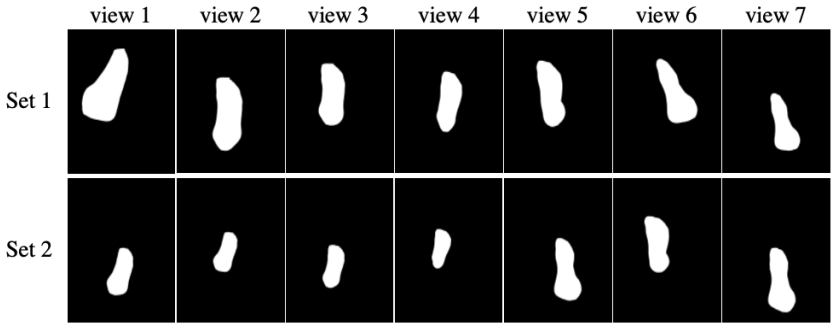


Fig. 9: Sample sets of synthetic silhouettes.

uniformly distributed from the inside to outside of the foot. The length and width are measured by hand. Camera pose is estimated using the AR features (visual inertial odometry) on current smart phones and foot segmentation is conducted using the pretrained deep neural network described previously.

6 Experiments

Evaluation metrics. For synthetic silhouettes where the ground truth 3D models of the feet are available, the 3D vertex error (Eq. 2) and width error, after scaling the vertices to a fixed foot length of 25 cm (roughly the length of an average foot), are used for evaluation of the shape performance. The width is defined as the *ball-joint width*, i.e., the 2D distance between the inner and outer extrema (Fig 2b). For the real image datasets, only the width error is used for shape performance evaluation since we do not have the ground truth 3D vertices. For scale, the results are reported using the l_1-loss between the predicted and ground truth lengths, for both real and synthetic data.

Baseline - optimisation method We compare our model to a traditional 3D optimisation approach of fitting the PCA model to the silhouette data. This approach aims to minimise the overlap of the model projected silhouette with

that of the predicted silhouette. We use a differentiable renderer [19] and a silhouette reprojection based on $L2$ distance between the model and predicted silhouettes. This loss function is optimised using Adam [30] jointly across all camera views.

6.1 Shape-only model

As an initial experiment, we first test our model for shape reconstruction by removing the scale prediction. This gives us insight into the model's ability to first obtain the correct shape, regardless of scale. The training loss is now only the shape loss L_{shape} (Eq. 4). Since the model is not regressing the foot scale, we centre and scale the foot in the silhouettes such that the foot size relative to the image is fixed. Max-pooling is used for combining the feature vectors from different viewpoints. For each training instance, we sample 3 random views of the same foot. The network is trained using Adam optimiser, batch size of 32, learning rate of 1×10^{-3} and $\delta_{shape} = 3$ mm. The steps per epoch is calculated such that the model on average "sees" every possible combination for every foot.

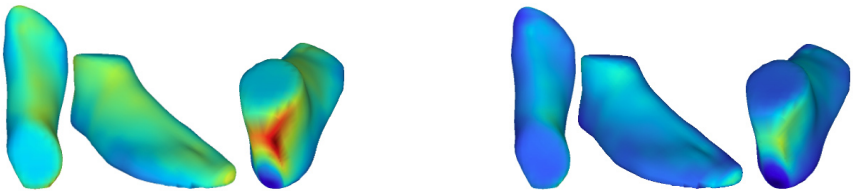
Table 2: Average vertex error and width error for the shape-only model from different number of views.

Number of views	Vertex error (mm)	Width error (mm)
1	0.9	0.9
2	0.6	0.5
3	0.6	0.4
4	0.5	0.4
5	0.5	0.4
6	0.5	0.4
7	0.5	0.4

On synthetic test data, Table 2 shows that the model achieves very small average errors (< 1 mm). For reference, if a model only predicts the mean foot shape, the average errors would be 3.7 mm and 2.6 mm for vertex and width respectively. Even though our model is trained on 3-view inputs, it generalises to other number of viewpoints and the performance generally improves as we increase the number of viewpoints. This is because it is less likely that a part of the foot is hidden when more views are given. As we further increase the number of views, the improvement in vertex accuracy decreases and it eventually converges to a vertex error of 0.5 mm.

The best width error is achieved with only 3 views; further increasing the number of views result in a similar or worse width error. This is because the model is trained to minimise the average vertex error and at any time the model does not know or care about the width error. As more views are given, the model is able to compute a set of PCA coefficients that results in a smaller vertex error, i.e., better overall shape reconstruction, but in return a slight amount of width accuracy is sacrificed.

We can see how the model makes use of different views to predict the foot shape by looking at how the vertex error is distributed across the foot. Fig 10 shows that with just a top-down silhouette, the model tends to produce larger errors around the foot dorsum (top surface of the foot) due to the lack of information of its shape. By incorporating more viewpoints, the overall error is reduced and more evenly distributed across the foot. For both types of input, relatively large error is made on toe and heel regions. This is because the vertices in these areas are more dense and have more variation across different feet.



(a) Single top-down input (b) Random 3-view input

Fig. 10: Vertex error distribution for the model tested on (a) single top-down silhouettes and (b) random 3-view silhouettes. The heatmaps have the same colorscale.

6.2 Full model (shape and scale)

We now include the length output unit and use the weighted loss (Eq. 6) as training loss. No processing (centering/scaling) is done to the silhouettes. The network is trained on 3-view inputs using Adam optimiser, batch size of 32, $w_{shape} = 1$, $w_{scale} = 0.05$, learning rate $= 1 \times 10^{-3}$, $\delta_{shape} = 3$ mm and $\delta_{scale} = 5$ mm.

Synthetic test data. On synthetic test data, our model achieves a vertex error of 1.2 mm, width error of 0.9 mm and length error of 2.1 mm. The length predictions are more accurate than the half-size increments (4.2 mm) in the UK and US shoe sizing systems. Note that there is an increase in the vertex error, from 0.6 mm with the shape-only model. This is mainly due to the fact that the foot is no longer centred in the silhouettes and the size of the foot relative to the image varies, making the task more difficult.

Real test data. To optimise the performance on real data, we split the 3-view real foot image dataset into validation and test sets, with 10 and 11 sets of images respectively, and retrained the network using the same set of hyperparameters except that the learning rate is reduced to 5×10^{-4} and early stopping is applied on the *real validation set*. We select the epoch that has the lowest sum of length error and width error on the real validation set as the final model. In addition, we address the domain gap through image augmentation including different degrees and types of blurring for estimating segmentation confidence

scores at silhouette periphery, and pixel noise to emulate incorrect segmentation. These augmentations are found to always improve the model performance on real data.

Table 3 shows that there is a large generalisation error going from synthetic to real data for all metrics, which could be due to noise in segmentation and camera pose. The best performance is achieved when the model is trained with max-pooling, and at test time we replace the max-pooling with average-pooling. On synthetic test data, however, doing the same thing would worsen the performance in both the length and shape prediction (length error: from 2.1 to 3.0 mm, vertex error: from 1.2 to 1.5 mm). For the case of synthetic silhouettes, Wiles and Zisserman [23] demonstrated that max-pooling outperforms average-pooling for combining the feature vectors as the former allows the combined features to be jointly recorded by different inputs views and important features from each viewpoint can be passed directly to the decoder so that the model can exploit information from specific views to reconstruct specific parts of the foot. With average-pooling, the important features are averaged out by other views and the model is forced to reconstruct every part of the foot with information from all given views. However for real data, average-pooling outperforms max-pooling because there is a lot more noise in the data (e.g. faulty segmentation/camera calibration). In this case, average-pooling helps reduce the effect of these errors by averaging them out, whereas max-pooling could allow noisy information to flow directly to the decoder.

There is also a big gap between the validation and test performance which is mainly due the the small amount of real data available (22 sets of images) and can be overcome by collecting a larger set of data so that the validation results are more reliable.

Table 3: Average length and width errors (mm) on real val/test data.

Trained with	Validation set		Tested with **max-pool**		Tested with **avg-pool**	
	Length error	Width error	Length error	Width error	Length error	Width error
max-pool	5.3	4.4	8.4	4.0	6.5	4.3
avg-pool	5.4	4.2	12.2	3.7	8.3	3.4

6.3 Effect of silhouette accuracy

To investigate the error introduced by the imperfect predicted segmentation, we manually segmented 4 sets of foot images from the 3-view real foot image dataset and tested our model (trained with max-pooling) on these ground truth silhouettes. Results are reported in Table 4 and sample reconstructions are displayed in Fig 11.

In terms of shape, we see from the sample reconstructions that the feet reconstructed using hand-segmented silhouettes are much closer to the actual foot because the shape prediction is heavily influenced by the input silhouettes. For both types of silhouette, the projections of the reconstructed feet fit almost perfectly to the input silhouettes so it is hard for the model to produce an accurate

shape prediction when the silhouettes do not represent the actual foot shape due to error by the segmentation network. In terms of length, hand-segmentation in fact worsens the model performance which suggests that there are error sources other than the segmentation network, such as noise in the camera poses and occlusion of the heel by the ankle or leg.

Table 4: Average length and width errors (mm) on 4 sets of test silhouettes for two different segmentation methods (segmentation network in deep network predicted and hand-segmentation), using the model trained with max-pooling.

Pool at test time	**Predicted segmentation**		**Hand segmentation**	
	Length error	Width error	Length error	Width error
max-pool	4.5	3.0	5.5	3.8
avg-pool	6.9	5.5	9.1	8.0

6.4 Number of views

We compare our model with the baseline method on the 20-view real foot image dataset, from 1 to 20 views. For each number of views, the viewpoints are chosen such that the view angles are approximately uniformly distributed. This allows us to analyse the stability of the methods to camera angles. It is found that to achieve good performance consistently, a uniform sample of 3 views is sufficinet for our model whereas the baseline requires at least 6. Table 5 records the mean errors from 6 views onwards which is when both methods have enough views to work at their best. On average, our network outperforms the optimiser in both length and width accuracy.

Table 5: Average length and width errors *from 6 to 20 views* on the 20-view real foot image dataset.

Method	Length error (mm)	Width error (mm)
FootNet (ours)	$\mathbf{4.3 \pm 0.4}$	$\mathbf{6.9 \pm 0.3}$
Optimisation	6.4 ± 1.9	7.9 ± 0.5

Prediction time. For one view, our model and the optimiser take 27 ms and 27000 ms respectively on average to make one reconstruction. For 20 views, our model and the optimiser on average take 435 ms and 98000 ms respectively. The optimiser requires GPUs for its computation whereas our model was tested on a CPU and is still 2 to 3 orders of magnitude faster as it only requires a single forward pass through the network. This shows that our network is capable of reconstructing the foot using only the computation power of a smart phone in a much quicker time.

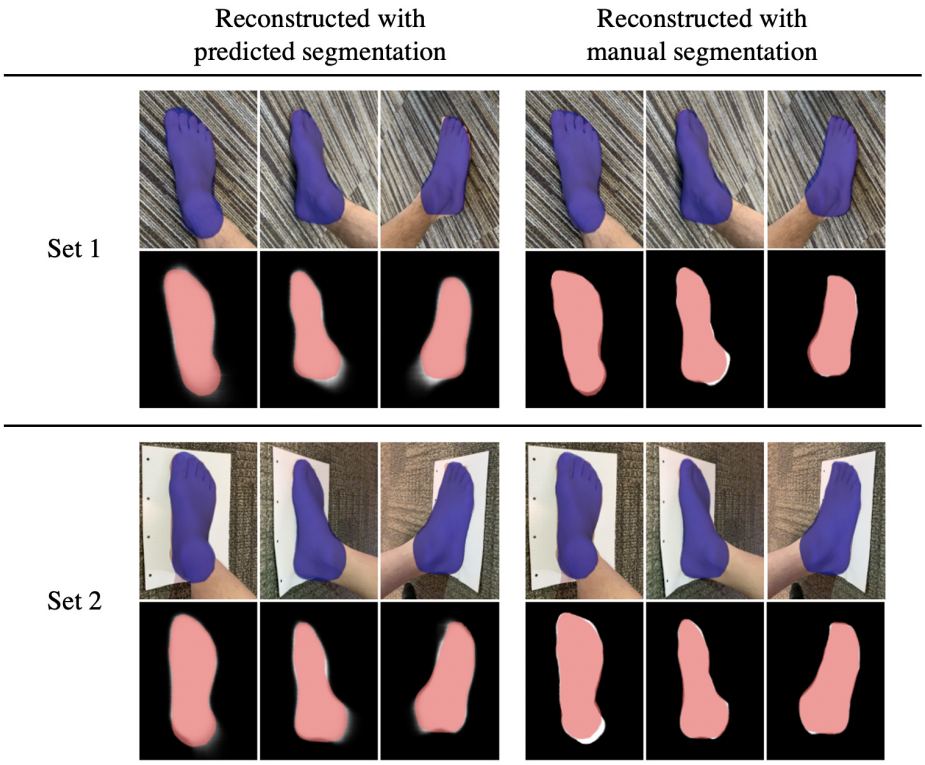


Fig. 11: Sample reconstructions from deep network predicted segmentation (left) and hand segmentation (right). The reconstructed feet are reprojected to the original image and the silhouettes for comparison.

7 Conclusion

The aim of this paper is to develop a framework that reconstructs the foot from 2D images on mobile devices. Using PCA, we compress the foot shape to only 10 parameters. We propose a CNN architecture that regresses the shape parameters given multiple foot silhouettes and corresponding camera poses. On synthetic silhouettes, the model achieves very high accuracy (vertex error close to 1mm and length error of 2mm) and we demonstrate that it generalises to different number of input views at test time. On real data our model outperforms a classical optimisation-based method both in accuracy and inference speed. Future work will involve improving generalisation of our method to real data.

Acknowledgments

We would like to thank Benjamin Biggs for providing the baseline optimisation code, Stefano Bucciarelli for help with the PCA model construction and TRYA SrL for providing a fraction of their dataset for ground truth 3D images of feet.

References

1. VoxelCare: 3D Laser Foot Scanner. (accessed 24/05/2020) https://www.voxelcare.com/#!/content/3D-Laser-Foot-Scanner.
2. Corporation, V.R.: Yeti 3D Scanner. (accessed 24/05/2020) https://vorum.com/yeti-3d-foot-scanner.
3. Moons, T., Van Gool, L., Vergauwen, M.: 3d reconstruction from multiple images: Part 1 - principles. Foundations and Trends in Computer Graphics and Vision **4** (2009) 287–404
4. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2 edn. Cambridge University Press, USA (2003)
5. Koyuncu, B., Kullu, K.: Development of an optical 3d scanner based on structured light. In: Proceedings of the 9th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases. (2010) 17–22
6. Precision3D: Fotoscan 3D Foot Scanner. (accessed 24/05/2020) http://www.precision3d.co.uk/fs.htm.
7. Kutulakos, K.N., Seitz, S.M.: A theory of shape by space carving. In: ICCV. (1999)
8. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: CVPR. (2017)
9. Tulsiani, S., Efros, A.A., Malik, J.: Multi-view consistency as supervisory signal for learning shape and pose prediction. In: CVPR. (2018)
10. Liu, S., Giles, L., Ororbia, A.: Learning a hierarchical latent-variable model of 3d shapes. In: 3DV. (2018)
11. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR. (2015)
12. Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., Tenenbaum, J.: Marrnet: 3d shape reconstruction via 2.5 d sketches. In: NIPS. (2017)
13. Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J.B., Freeman, W.T.: Pix3d: Dataset and methods for single-image 3d shape modeling. In: CVPR. (2018)
14. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions on Graphics **36** (2017) 1–11
15. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: ICCV. (2017)
16. Dai, A., Ruizhongtai Qi, C., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: CVPR. (2017)
17. Cao, Y.P., Liu, Z.N., Kuang, Z.F., Kobbelt, L., Hu, S.M.: Learning to reconstruct high-quality 3d shapes with cascaded fully convolutional networks. In: ECCV. (2018)
18. Yang, B., Rosa, S., Markham, A., Trigoni, N., Wen, H.: Dense 3d object reconstruction from a single depth view. IEEE transactions on pattern analysis and machine intelligence **41** (2018) 2820–2834
19. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR. (2018)
20. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: ECCV. (2018)
21. Dibra, E., Jain, H., Öztireli, C., Ziegler, R., Gross, M.: Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks. In: 3DV. (2016)
22. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV. (2016)

23. Wiles, O., Zisserman, A.: Silnet : Single- and multi-view reconstruction by learning from silhouettes. In: BMVC. (2017)
24. Amstutz, E., Teshima, T., Kimura, M., Mochimaru, M., Saito, H.: Pca based 3d shape reconstruction of human foot using multiple viewpoint cameras. In: ICCV. (2008)
25. Lunscher, N., Zelek, J.: Point cloud completion of foot shape from a single depth map for fit matching using deep learning view synthesis. In: ICCV Workshop. (2017) 2300–2305
26. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
27. Campbell, N.D., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multi-view stereo. In: ECCV. (2008)
28. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
29. Huber, P.J.: Robust estimation of a location parameter. Ann. Math. Statist. **35** (1964) 73–101
30. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)