

# APPENDIX

## A: Network Architectures

**Entangled UnshearNet:** The encoder  $E$  compresses the input  $256 \times 256$ -pixel tactile image using five convolutional (Conv) layers, each followed by batch normalization (BN) and rectified linear unit (ReLU) activation layers respectively (architecture in Fig. 2). The output of the last Conv layer ( $8 \times 8 \times 64$ ) is passed to the decoder  $D$  which upsamples it to an output  $256 \times 256$ -pixel canonical image  $PC$ . Similarly to the Conv layers, all transposed convolutional layers (T-Conv) were followed by BN and ReLU activation layers except the output layer which used sigmoid activation function instead (see also Fig. 2).

This model was used as a baseline to evaluate the performance of proposed *Disentangled UnshearNet* model on its effectiveness in removing distortion in sensor response caused by sliding-induced global shear. For details of the training, please see Appendix B.

**Disentangled UnshearNet:** The encoder has the same inputs and architecture as the *Entangled UnshearNet*, except the the output of the 5<sup>th</sup> Conv layer is followed by two additional Conv layers, one each for the two latent codes: Pose ( $8 \times 8 \times 64$ ) and Shear ( $8 \times 8 \times 64$ ).

One decoder  $DC$  takes as input pose latent code and upsamples it to  $256 \times 256$ -pixel canonical output tactile image  $PC$ . All T-Conv layers were followed by BN and ReLU activation layers except the output layer which used sigmoid activation layer instead (architecture shown in Fig. 3).

The other decoder  $DS$  takes as input both pose and shear codes, merges them and upsamples to an output  $256 \times 256$ -pixel sheared image  $PS$  (intended to match the encoder input). Apart from the above differences,  $DS$  has the same architectures as  $DC$ . For details of the training, please see Appendix B.

**PoseNet:** This model takes as input  $256 \times 256$  image, compresses it to extract features using the Conv part which are then combined using fully connected (FC) part to predict continuous-value pose components at the output. In total, the network consists of five convolution layers, two max pooling (MP) layers following 2<sup>nd</sup> and 4<sup>th</sup> Conv layers; the CONV layers were followed by BN and ReLU activation layers respectively. The 1<sup>st</sup> FC layer used ReLU activation while the output FC layer used sigmoid activation. For details of the architecture and training see Fig. 5 and Appendix B.

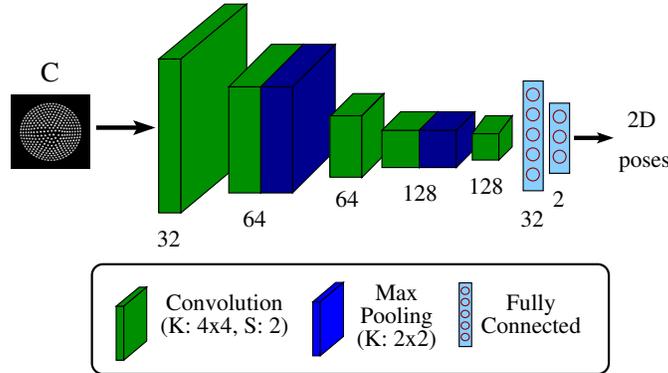


Figure 5: **PoseNet schematic:** This network takes as input, a tapping binary sensor image,  $C$ . The convolution part compresses input to extract relevant features which are then combined by fully connected part to predict continuous-valued object pose parameters at output.

## B: Training Details

All the inputs (binary  $256 \times 256$ -pixel images) and outputs (binary  $256 \times 256$ -pixel images for image-to-image models, pose parameters for image-to-pose model) were scaled to the range  $[0, 1]$ . All networks weights were initialized from a zero-centered normal distribution with 0.02 standard deviation. Both image-to-image models were trained on data collected from all three stimuli shapes (Fig. 1 (c)) as discussed in section 3.2.

For image-to-image models, we used a batch size of 32. All convolutional/transposed convolutional layers used L1/L2 regularization ( $10e-4$ ) along with random image shifts, 1-2% of image size to prevent overfitting. The  $L2$  loss computed across the entire image and  $L1$  patch loss – computed between 100 random crops of size  $20 \times 20$  of generated unshered images and corresponding canonical images – were used to train the networks in the ratio 10:1. In case of *Disentangled UnshearNet*, the encoder (E) and shear decoder (DS) were trained using reconstruction loss.

For image-to-pose model, we used a batch size of 256. Like image-to-image models, all layers used L1/L2 regularization ( $10e-4$ ) along with random image shifts, 1-2% of image size to prevent overfitting. The network was trained via the  $L2$  loss computed between predicted and target pose parameters.

We used ADAM optimizer [23] with learning rate of 0.0001,  $\beta_1 = 0.5$  and  $\beta_2 = 0.999$ . We used learning rate scheduling for training. For image-to-image models, the learning rate was reduced to one-fifth and one-tenth after 40 and 80 epochs. The models were trained, in total, for 100 epochs. For image-to-pose model, the learning rate was reduced to half and one-tenth after 100 and 200 epochs respectively. The model was trained for 250 epochs. For all models, the model with best validation accuracy was used for testing.

Finally, training and optimization of the networks was implemented in the Tensorflow 2.0 library on a NVIDIA GTX 1660 (6 GB memory) hosted on a Ubuntu machine.

## C: Control Policy for Continuous Contour Following

Local pose estimation allows a robot to maintain contact while safely moving over the object, thus enabling complex robot-object interactions. To demonstrate continuous 2D contour following, we used a simple control policy with following two aims: 1) keep the sensor normal to object surface while in motion and 2) at every time step  $t$ , move the sensor tangentially along the surface by a predefined step (0.5 mm in this case). To achieve these aims, a discrete-time proportional-integral (PI) controller was implemented to output a change in the pose of the sensor ( $\Delta p(t)$ ) in its reference frame

$$\Delta p(t) = K_p e(t) + K_i \sum_{t'=0}^t e(t')$$

where  $K_p$  and  $K_i$  are diagonal gain matrices with proportional gain of 0.5 and integral gains of 0.3 and 0.1 for translations and rotations respectively.  $e$  was error between the predicted pose and a reference normal to the edge.

## D: Disentanglement of Latent Representations

An ablation study was used to verify the separation of latent representations in the *Disentangled UnshearNet* into pose and shear codes respectively. To do this, we passed the ‘Shear Code’ to the unsheared reconstruction decoder (DC) instead of the ‘Pose Code’. As expected, this led to a severe degradation in performance with mean-square error (MSE) between the ground truth images  $C$  (tap) and unsheared images  $PC$  increasing to 0.22, an order of magnitude higher than original of 0.023. In similar fashion, the SSIM index dropped to 2% when using the Shear Code to reconstruct the unsheared images  $PC$  instead of the original 93% when the Pose Code was used. Likewise, a similar degradation was observed when the ‘Shear Code’ was replaced by the Pose Code to reconstruct the sheared input  $S$ . The mean-squared error between  $S$  and the sheared output  $PS$  increased to 0.1, which was 50-times the original of 0.002 when both ‘Pose and Shear Codes were used to reconstruct  $PS$ . The SSIM index showed a similar trend with the similarity dropping from 99.5% to only 11%. This shows that the Shear Code is indeed encoding relevant information required for successful reconstruction of sheared input  $S$ . These results clearly demonstrate that the *Disentangled UnshearNet* successfully disentangles the latent representations as desired.

### E: High Resolution Results

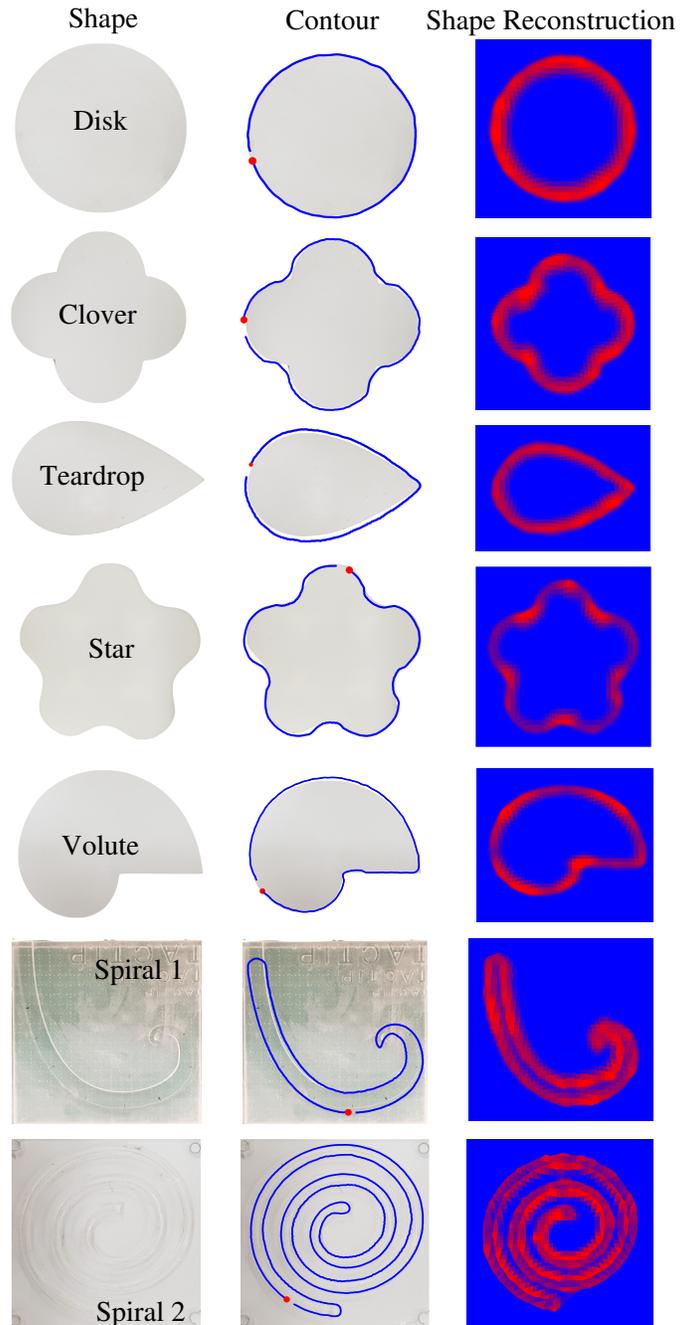


Figure 6: **Contour following & Shape Reconstruction.** Left: Stimuli shapes, Middle: Robust sliding across shapes in left row post removal of shear using *Disentangled UnshearNet*. Red dot shows the starting point/initial contact. Right: Full shape reconstruction post removal of shear using *Disentangled UnshearNet*.