Bodies at Rest: 3D Human Pose and Shape Estimation from a Pressure Image using Synthetic Data Supplementary Material

Henry M. Clever[1], Zackory Erickson[1], Ariel Kapusta[1], Greg Turk[1], C. Karen Liu[2], and Charles C. Kemp[1]

[1]Georgia Institute of Technology, Atlanta, GA, USA, [2]Stanford University, Stanford, CA, USA

{henryclever, zackory, akapusta}@gatech.edu, turk@cc.gatech.edu, karenliu@cs.stanford.edu, charlie.kemp@bme.gatech.edu

Appendix A: PressurePose Data Generation

A.1. Initial Pose Sampling

We use rejection sampling to generate initial pose dataset partitions. Our criteria are as follows.

Uniform Cartesian space distribution - Fig. 10 (a). We use rejection sampling to uniformly sample poses with respect to the Cartesian space, by discretizing the space and ensuring that a given limb is equally represented in each unit. We define a Cartesian space $\boldsymbol{\mathcal{Y}}$ as a cuboid for checking for presence of the most distal limb. First, we constrain $\boldsymbol{\mathcal{Y}}$ in the (x, y) directions to how far the distal joint (e.g. right foot, $\boldsymbol{s}_{r.foot}$) can extend from the promixal joint (e.g. right hip, $\boldsymbol{s}_{r.hip}$) in a limb. For the legs, we assume that the foot cannot move above the hip. For the right leg, these constraints can be summarized as: $s_{r.foot,x} \in [s_{r.hip,x} - l_{leg}, s_{r.hip,x} + l_{leg}]$ and $s_{r.foot,y} \in [s_{r.hip,y}, s_{r.hip,y} + l_{leg}]$. We also constrain the z direction to ensure that the distal joint is initially positioned at a height close to where the proximal joint is: For laying poses, the distal joints (feet and hands) are more likely to end up close to the surface of the bed than very high in the air, for example. This constraint promotes simulation stability and decreases the time it takes for physics simulation #1 (Fig. 2) to reach an equilibrium state. We constrain $s_{r.foot,z} \in [s_{r.hip,z} - 10cm, s_{r.hip,z} + 10cm]$.

Next, we break up $\boldsymbol{\mathcal{Y}}$ into a set of smaller cuboids as shown in Fig. 10-top middle. For each limb we uniformly sample a cuboid from $\{\mathcal{Y}_1, ...\}$ and then use rejection sampling on the limb joint angles — in the case of Fig. 10 (a), the right leg — until $\boldsymbol{s}_{r.foot} \in \mathcal{Y}_4$.

Generate common posture partitions - Fig. 10 (b). Some common postures, such as resting with the hands behind the head, are unlikely to be generated when the joint angles are sampled from a uniform distribution. For example, there is a $< 1\%$ probability of generating a pose with the hands-behind-the-head when sampling joint angles uniformly, so a network trained with such little hands-behind-the-head data has difficulty learning such a pose. We mitigate this issue by checking for presence of the most distal joint in a cuboid representing where it would be located in such as pose. If the joint is within the cuboid, e.g. $\boldsymbol{s}_{r.hand} \in \mathcal{Y}_{RH}$, the joint passes the criteria and we add the limb pose to the set of checked initial poses.

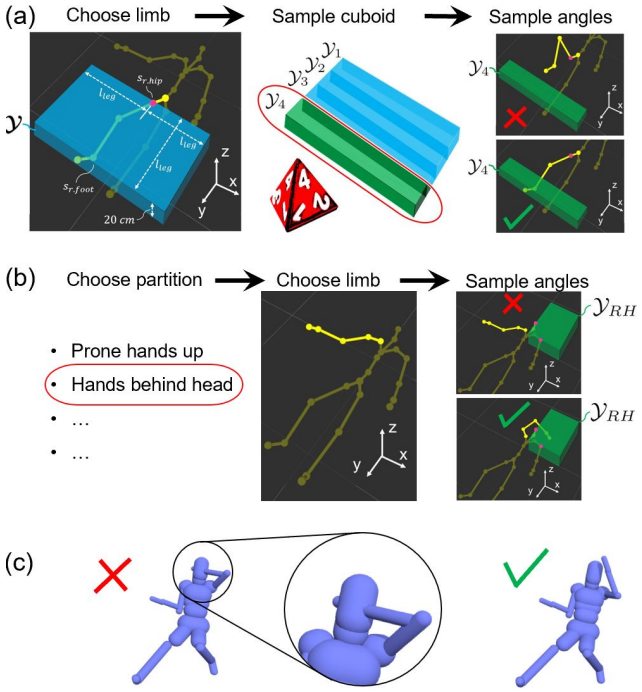

Figure 10. Rejection sampling criteria. (a) Evenly distributing right leg poses across Cartesian space by sampling from four non-overlapping Cartesian cuboids, $\{\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3, \mathcal{Y}_4, \} \in \boldsymbol{\mathcal{Y}}$. Reject pose angles if $\boldsymbol{s}_{r.ankle} \notin \mathcal{Y}_4$ (b) For sampling right arm in the hands-behind-head partition, we reject the right arm pose angles if $\boldsymbol{s}_{r.hand} \notin \mathcal{Y}_{RH}$. (c) Pose feasibility checking via collision detection.

Prevent self-collision - Fig. 10 (c). We reject poses that result in self collision by capsulizing the mesh and using the DART collision detector. We check the hands, forearms,

feet, and lower leg capsules for collision with any other capsules except their adjacent capsules (e.g. forearm and upper arm should overlap).

A.2. Dynamic Simulation Details

Weighting particles in FleX. We directly calculate particle mass for the particlized human in physics simulation #2, as well as for the particlized calibration objects depicted in Fig. 5 (b). Since FleX is a position-based dynamics simulator and the mass is defined by units of inverse mass $1/m$ on an arbitrary scale, we begin by defining the inverse mass scale for particles in the particlized human.

For this, we assume that the volume each particle in the human takes up, as well as the density of particles, is the same for that of water. Because volume and density are equal, we also can set inverse mass equal, so $1/m_H = 1$, thus $1/m_{H2O} = 1$.

We calculate the inverse mass for particles in calibration objects by a density ratio to that of water, given a known weight of the object w_k and the object volume V_o:

$$\frac{1}{m_{o,k}} = \frac{1}{m_{H2O}} \frac{\rho_{H2O}}{\rho_o} = \frac{\rho_{H2O}}{m_{H2O}} \frac{V_o}{w_k/g} \tag{10}$$

where ρ_{H2O} is the density of water and g is gravity. In contrast to the humans and objects rested on the bed, the the soft mattress and synthetic pressure mat particle inverse mass are determined from an optimization described in Appendix A.4.

Weighting the capsulized human chain. We compute a per-capsule weight for the articulated capsulized chain in DartFleX based on the weight distribution for an average person and capsule volume ratios. First, we describe how we assign capsule mass for the average person. We use average body mass and mass distribution values from Tozeren [13], and calculate capsule volumes from body shape. We assume the average human of gender $g \in \{M, F\}$ has a mass of $\bar{m}_g$, mass percentage distribution for body part R of $\bar{X}_{R,g} \in \bar{\boldsymbol{X}}_g$, and SMPL body shape parameters $\bar{\boldsymbol{\beta}}_g = \mathbf{0}$. We define the mass of each capsule c in an average person to be:

$$\bar{m}_c = \bar{m}_g \bar{X}_{R,g} \frac{\bar{V}_{c,g}}{\bar{V}_{R,g}} \tag{11}$$

where $\bar{V}_{c,g}$ is the volume of capsule c for a mean body shape $\bar{\boldsymbol{\beta}}_g$, and $\bar{V}_{R,g}$ is the sum of volumes for all capsules in body part R. Now, we describe how this capsule mass can be converted into masses for people of other shapes. To find the mass of some capsule c for a body of particular shape $\boldsymbol{\beta}$, we use a capsule volume ratio between the particular person and an average person:

$$m_c = \bar{m}_c \frac{V_c}{\bar{V}_{c,g}} \tag{12}$$

where V_c is the volume of some arbitrary capsule. Computing capsule volume analytically is simple given radius and length, but this is complicated by capsule overlap, which is often substantial in the SMPLIFY capsulized model [2] we use. Instead, we use discretization to compute capsule volume and correct for overlap. First, we use the SMPLIFY regressor to calculate capsule radius and length from body shape $\boldsymbol{\beta}$. Besides shape, overlap is dependent on the particular pose of the capsulized model. We assume that pose dependent differences in overlap are very small, and set the pose constant at $\boldsymbol{\Theta} = \mathbf{0}$. We then compute the global transform for each capsule using this shape and pose. From capsule radii, lengths, and global transforms, we place all capsules in 3D space and voxelize them with a resolution of $2mm$. This produces a set of 3D masks, which are tagged to their corresponding capsules. Voxels belonging to a unique capsule are allocated directly, while voxels belonging to multiple capsules are allocated fractionally based on the number of capsules sharing the voxel. We compute capsule mass inertia matrices analytically from capsule radius and length.

Capsulized body joint stiffness. For an average person, we set the following joint stiffnesses for the shoulders, elbows, hands, hips, knees and feet to low stiffness: $\bar{k}_{\theta,shd} = 4$ Nm, $\bar{k}_{\theta,elb} = 2$ Nm, $\bar{k}_{\theta,hnd} = 4$ Nm, $\bar{k}_{\theta,hip} = 6$ Nm, $\bar{k}_{\theta,knee} = 3$ Nm and $\bar{k}_{\theta,feet} = 6$ Nm. We set torso and head stiffness very stiff $\bar{k}_{\theta,trs}, \bar{k}_{\theta,hd} = 200$ Nm. For a person of particular body shape, we weight joint stiffnesses $\boldsymbol{k}_\theta$ by the body mass ratio, where $\boldsymbol{k}_\theta = (m/\bar{m})\bar{\boldsymbol{k}}_\theta$. We set joint damping $\boldsymbol{b}_\theta = 15\boldsymbol{k}_\theta$. The direction and magnitude of stiffness force on each joint is dependent on joint equilibrium position, i.e. the joint angle where force is 0. We set the equilibrium position of the joints to be the *home pose*, where the arms are at the sides and the legs are straight. In the SMPLIFY model, *home pose* consists of equilibrium joint positions $\boldsymbol{\Theta}_{eq}$ set to 0, except the shoulders, which are bent downward at 90 degrees. Rather than set $\boldsymbol{\Theta}_{eq}$ to initial joint angles $\boldsymbol{\Theta}_C$, we do this to guide the pose away from extreme angles at a modest force.

Because we set the joint stiffness low, our dataset does not capture non-resting postures, such when a person is getting in/out of bed (recall Table 1). However, we have been able to generate resting sitting poses by bending the mattress and pressure mat into a sitting configuration and then resting a person on it, like the sitting postures in [4].

Settling criteria - Physics simulation #1. For physics simulation #1, the goal is to slowly allow the body to fall on the bed and settle into a resting pose. We start the capsulized body at a height based on the lowest point on the body. For many randomly sampled poses, the lowest joint is initially much lower than the center of mass, which causes the center of mass to build significant momentum by the time it reaches the bed. We found that this caused bounc-

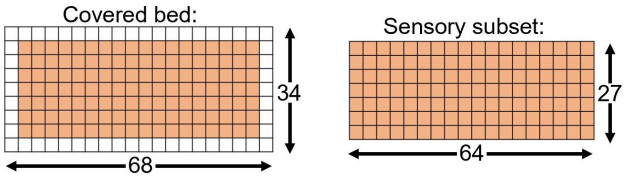

Figure 11. Size of synthetic pressure mat. Physics simulation #1 uses forces from particles on the entire covered bed. The pressure mat calculated in physics simulation #2 uses a smaller subset representing the size of the real pressure mat.

ing and instability, and was qualitatively different from the motion one might take to assume a resting pose in bed. We alleviate this issue by zeroing the velocity of the capsulized model every 4 iterations in the simulation ($\sim 0.04s$) until a capsule that better represents the center of mass contacts the surface of the bed. For this, we use the capsule approximating the buttocks.

Finding a resting pose in static equilibrium is hampered by the stability of DartFleX: DART uses a more traditional physics solver and FleX uses position-based dynamics, which are challenging to connect in a stable loop. Rather than run the simulation until static equilibrium, we use a cutoff threshold that takes velocity and acceleration of all capsules into account. We define a resting body as that when the maximum velocity of all capsules has reached $v_{max} < 0.05m/s$ and maximum acceleration has reached $a_{max} < 0.5m/s^2$. In the event the model does not settle within 2000 iterations or the pressure array becomes unstable (defined by separation of particles in the pressure mat, e.g. limb poking into mat), the simulation is terminated and the particular $\mathbf{\Theta}_C$ is rejected. Across the whole dataset, we found roughly a 10% rejection rate for both of these criteria.

Settling criteria - Physics simulation #2. We use the same approach as simulation #1 to determine the height to drop particlized humans. We found it to always be stable for our purpose, and it took roughly 150 iterations to reach the same resting velocity and acceleration previously stated. Because it only uses FleX and the limbs do not move kinematically, it is an order of magnitude faster to run and provides greater flexibility to determine settling criteria. We ran simulation #2 for a minimum of 200 iterations and terminated it once the velocity and acceleration thresholds of the particlized human, $v_{ptcl} < 0.05m/s$ and $a_{ptcl} < 0.5m/s^2$, were reached. In almost all cases, 200 iterations was sufficient.

Computation time. For both physics simulations, we ran 10 parallel simulation environments on a computer with 32 cores and a NVIDIA 1070-Ti GPU. This allowed us to generate roughly 35,000 labeled synthetic pressure images per day.

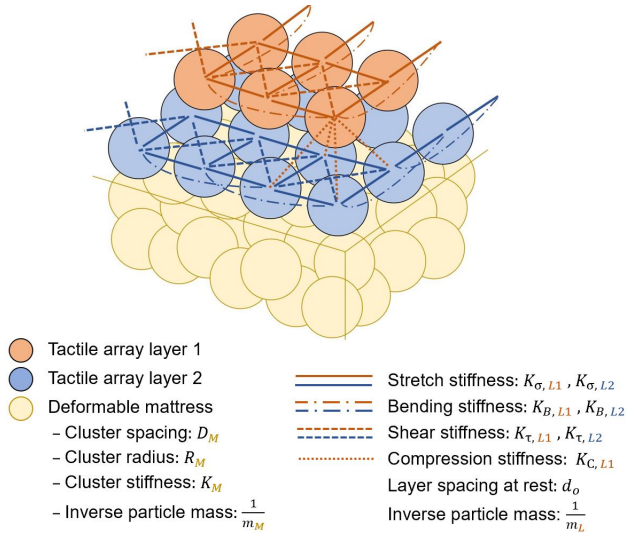

Figure 12. Pressure mat pyramidal structure showing FleX parameters that we optimized using CMA-ES.

A.3. Pressure Mat Structure Details

Limited pressure sensing area. The sensing portion of the real pressure mat does not cover the entire mattress. We measured a non-sensing border of 6 cm on the sides of the bed and 9 cm at the top and bottom. We built the simulator in the same way: the synthetic pressure mat covers the entire bed (68 x 33), but only an inner subset (64 x 27) representing the sensing area of the pressure image array is recorded, as depicted in Fig. 11.

FleX spring constraints. FleX particles in the synthetic pressure mat are bound together by stiffnesses shown in Fig. 12.

Pressure mat adhesion. For the real pressure mat, velcro and tape are used to prevent sliding across the bed. For the synthetic pressure mat, particles are fixed in horizontal directions across the bed.

A.4. FleX Calibration

Although FleX is able to simulate soft bodies, FleX is not optimized to model real-world physics or to calculate realistic pressures. To optimize our FleX simulation to match the real-world mattress and pressure mat, we place a set of static objects on the real mattress, and record the resulting pressure images from the pressure mat. We then build a similar environment in FleX, and we optimize FleX parameters such that the simulated and real-world measurements closely align.

We jointly optimize 16 deformable bed and pressure sensing array parameters $\mathcal{S}$ using CMA-ES [5]. These include the 13 FleX parameters in Fig. 12, including 4 soft mattress parameters, 7 pressure array stiffnesses, spacing between the pressure mat layers and particle inverse mass,

as well as quadratic taxel force constants C_1, C_2, and C_3. To optimize, we first place a set of real rigid objects $\{\mathfrak{o}_1, \dots \mathfrak{o}_J\}$ each with weights $\{w_1, \dots w_M\}$ on the real bed. Fig. 5 (a) depicts $\{\mathfrak{o}_1, \dots \mathfrak{o}_J\}$, where $J = 4$ and we use capsular objects with 5 weights for each: 1.3, 2.3, 4.5, 9.1 and 14 kg on the shorter capsules (L=20 cm), and 1.3, 4.5, 9.1, 14 and 18 kg on the longer capsules (L=40 cm). We then collect real pressure mat images $\{\mathbb{P}_{1,1}, \dots \mathbb{P}_{J,M}\}$ and measure the distance that the mattress compresses normal to the bed surface in centimeters, $\{\mathfrak{q}_{1,1}, \dots \mathfrak{q}_{J,M}\}$.

Next, we build a matching set of simulated capsules $\{o_1, \dots o_J\}$ in FleX with the same weights, where one of these objects is shown Fig. 5 (b). At each iteration of the optimization, we drop J simulated capsules of each M weights onto the FleX mattress, re-compute the synthetic pressure images, and compare them to the real ones. The loss function for our optimization takes as input simulated and real pressure images and is computed as:

$$\arg\min_{\mathcal{S}} \sum_{o=1}^{J} \sum_{k=1}^{M} \left(\mathcal{L}_{k,o}^{F} + \mathcal{L}_{k,o}^{C} + \mathcal{L}_{k,o}^{Q} \right) \tag{13}$$

with terms for force error in the pressure mat, $\mathcal{L}_{k,o}^{F}$, contact locations on the pressure mat, $\mathcal{L}_{k,o}^{C}$, and amount of mattress compression by the object, $\mathcal{L}_{k,o}^{Q}$. For some real object $\mathfrak{o}$ with weight k resting on a soft bed at depth $\mathfrak{q}$ from the unweighted height of the soft bed, a pressure image $\mathbb{P}$ measures forces on individual taxels $\{\mathfrak{u}_1 \dots \mathfrak{u}_T\}$, where contact is a binary vector $\{\mathfrak{c}_1 \dots \mathfrak{c}_T\}$ indicating which taxels are measuring non-zero forces. The upper limit T is a spatial index indicating the number of taxels on the pressure image. We note that the value of T for these calibration images is roughly equal to a fraction of the pressure mat size, $(64 \times 27)/5$, because we drop multiple objects simultaneously to speed up the optimization. Similar to the real mat, the values for the simulated environment are computed as u_i, c_i, and q. The loss terms are computed as:

$$\mathcal{L}_{k,o}^{F} = \frac{1}{2} \frac{\sum_{i=1}^{T} |u_i - \mathfrak{u}_i|}{\sum_{i=1}^{T} (u_i + \mathfrak{u}_i)} + \frac{1}{2} \frac{|\sum_{i=1}^{T} (u_i - \mathfrak{u}_i)|}{\sum_{i=1}^{T} (u_i + \mathfrak{u}_i)} \tag{14}$$

$$\mathcal{L}_{k,o}^{C} = \frac{1}{2} \frac{\sum_{i=1}^{T} |c_i - \mathfrak{c}_i|}{\sum_{i=1}^{T} (c_i + \mathfrak{c}_i)} + \frac{1}{2} \frac{|\sum_{i=1}^{T} (c_i - \mathfrak{c}_i)|}{\sum_{i=1}^{T} (c_i + \mathfrak{c}_i)} \tag{15}$$

$$\mathcal{L}_{k,o}^{Q} = \frac{|q - \mathfrak{q}|}{|q| + |\mathfrak{q}|} \tag{16}$$

The first term for both $\mathcal{L}_{k,o}^{F}$ and $\mathcal{L}_{k,o}^{C}$ account for errors in pressure measurements between individual taxels between the real and simulated pressure mats. The second term accounts for errors in the total measured pressure under an object. All terms are normalized. Since the distances q and $\mathfrak{q}$ are signed, we take the absolute value in the denominator of $\mathcal{L}_{k,o}^{Q}$ for normalization.

CMA-ES implementation. To optimize the FleX environment with CMA-ES [5], we used a population size of 50, max iterations of 3000, max function evaluations of $1e+8$, mean learning rate of 0.25, function tolerance of $1e-3$, function history tolerance of $1e-12$, x-change tolerance of $5e-4$, max standard deviation of 4.0, and stagnation tolerance of 100. We used a machine with 8 cores and a Nvidia 1070-Ti GPU, and the optimization took 6 days.

Various combinations of parameters result in simulation instability. We perform a constrained optimization by placing a high cost on the evaluation function, `f_eval`, when a parameter is suspected of causing instability.

- Negative FleX parameters can cause instability. If any negative FleX parameter is proposed, a high `f_eval` is assigned.
- Large differences between K_σ, K_B, K_τ (see Fig. 12) causes knotting in the simulated array. If any stretch, bending, or shear stiffness value is outside of the range $0.5 < K < 2.5$, we add $10x$ the deviation from this range to the `f_eval`.
- An unusually long simulation time step indicates instability in the parameters. In this event, the particular rollout is terminated and a high `f_eval` is assigned.
- If an object takes too long to settle, the rollout is terminated and a high `f_eval` is assigned.

A.5. DartFleX Calibration

The purpose of this calibration is to calibrate the force that should be applied to a DART capsule from particle penetration on the FleX pressure mat. This enables the two simulators to be connected through a mass-spring-damper model, which we described in Section 3.2 in the main paper.

We begin with an optimized FleX environment (Appendix A.4) and calibrate the spring coefficient k, from the mass-spring-damper model. We calibrate k so that the *dynamic* collision geometries displace the FleX mattress in the same way that real objects would. We take the same set of real objects from the FleX calibration of various shapes $\{\mathfrak{o}_1, \dots \mathfrak{o}_D\}$ and weights $\{w_1, \dots w_D\}$, where $D = 20$, place them on the real mattress, and measure the mattress displacement $\{\mathfrak{q}_1, \dots \mathfrak{q}_D\}$. Then, we recreate the objects as collision geometries $\{\tilde{o}_1, \dots \tilde{o}_D\}$ in FleX, displace the FleX mattress by $\{\tilde{q}_1, \dots \tilde{q}_D, \} = \{\mathfrak{q}_1, \dots \mathfrak{q}_D\}$, and record the sum of particle penetration distances of underlying taxels $\left\{ \sum_{i=1}^{P} \boldsymbol{x}_{i,1}, \dots \sum_{i=1}^{P} \boldsymbol{x}_{i,D} \right\}$. We compute k as the average k across D objects:

$$k = \left(\frac{w_1}{\sum_{i=1}^{P} \boldsymbol{x}_{i,1}} \bigg|_{\tilde{q}_1} + \dots + \frac{w_D}{\sum_{i=1}^{P} \boldsymbol{x}_{i,D}} \bigg|_{\tilde{q}_D} \right) / D \tag{17}$$

where the vertical bar indicates the amount that object $\tilde{o}$ of weight w is displaced by distance $\tilde{q}$, which results in particle penetration distances $\sum_{i=1}^{P} \boldsymbol{x}_i$. The length of a timestep is uncontrollable in FleX. Thus, the timestep in DART is calculated by dropping objects in both environments from a matching height and equating the time to contact the ground, where both simulators have $g = 9.81m/s^2$. This resulted in a DART timestep of $0.0103s$.

A.6. Real Dataset Collection Details

Participants donned an Optitrak motion capture suit with high contrast to the bed sheets to facilitate analysis of the pose and body shape. We provided S, M, L and XL sizes, and instructed participants to use a form fitting size.

We used the IAI Kinect2 package to calibrate the Kinect [14]. Our released dataset consists of RGB images and depth/point cloud data from the Kinect that are synchronized and spatially co-registered to the pressure images. We manually synchronized the modalities; only static poses are captured so the time discrepancy is insignificant. We spatially co-registered the Kinect to the pressure mat by putting 1" tungsten cubes on the corners of the pressure mat, which could be seen with all modalities. We captured a co-registration snapshot for each participant, which was taken after they were finished. We created an interface to click on the tungsten block locations on the images and used CMA-ES to find the 6DOF camera pose and co-register it with the mat.

A.7. Dataset Partitions

Table 4 presents a detailed description of the data partitions. We split the data for gender. We also split for requiring initial limb positions to be over the surface of the bed, meaning that the Cartesian cuboids used for initial pose sampling (recall Fig. 10) are clipped in the x and y directions at the edge of the mattress.

A.8. Dataset Limitations

Domain gap. The real pressure mat has a larger force range. Additionally, as a result of putting a blanket on the bed during the real study, the overall pressure magnitude was reduced $\sim 3x$, which was not reflected in synthetic data calibration. To correct for this, we normalize as described in Appendix B.1.

Synthetic body joint limits. We observed that roughly 2% of the synthetic poses appear uncomfortable or infeasible for a real person (Fig 13). This work could be improved by using pose-conditioned joint angle limits such as [1] instead of constant limits. Fig. 13-right shows an impossible pose where the thighs are in collision. We were not able to check collisions between the thighs using the capsulized model because the thigh capsules are often in collision for valid poses.

pose partition, limb distribution	gender	limbs on bed	train ct. synth	test ct. synth	test ct. real
general*	F	N	26000	3000	120
even leg space: $\{\mathcal{Y}_1, ...\mathcal{Y}_4\} \in \boldsymbol{\mathcal{Y}}_L$	M	N	26000	3000	119
even arm space: $\{\mathcal{Y}_1, ...\mathcal{Y}_8\} \in \boldsymbol{\mathcal{Y}}_A$	F	Y	26000	3000	120
	M	Y	26000	3000	120
supine general**	F	N	13000	1500	40
even leg space: $\{\mathcal{Y}_1, ...\mathcal{Y}_4\} \in \boldsymbol{\mathcal{Y}}_L$	M	N	13000	1500	39
even arm space: $\{\mathcal{Y}_1, ...\mathcal{Y}_8\} \in \boldsymbol{\mathcal{Y}}_A$	F	Y	13000	1500	40
	M	Y	13000	1500	40
supine hands behind head**	F	Y	2000	500	40
even leg space, arms Fig. 10(b)	M	Y	2000	500	40
prone hands up†	F	Y	4000	500	40
even leg space, hnds above shldrs	M	Y	4000	500	40
supine crossed legs**	F	N	2000	-	-
even leg space, even arm space,	M	N	2000	-	-
feet must cross according to	F	Y	2000	500	40
x direction in Fig. 10(a)	M	Y	2000	500	38
supine straight limbs**	F	N	2000	-	-
even leg space, even arm space,	M	N	2000	-	-
elbows and knees straight	F	Y	2000	500	40
	M	Y	2000	500	36
TOTAL	-	-	184000	22000	952

Table 4. Partitions for synthetic data and prescribed poses. For evening the leg space, see Fig. 10(a). For evening the arm space, an additional four subspaces $\{\mathcal{Y}_5, \ldots \mathcal{Y}_8\}$ are chosen because the most distal joint (hand) is allowed to extend all the way below and above the limb root joint (shoulder), measured in the y direction.
* $\theta_{r,3} \sim \mathcal{U}[-\frac{\pi}{3}, \frac{\pi}{3}], \theta_{r,1} \sim \mathcal{U}[-\pi, \pi]$
** $\theta_{r,3} \sim \mathcal{U}[-\frac{\pi}{3}, \frac{\pi}{3}], \theta_{r,1} = 0$
† $\theta_{r,3} \sim \mathcal{U}[-\frac{\pi}{3}, \frac{\pi}{3}], \theta_{r,1} = \pi$

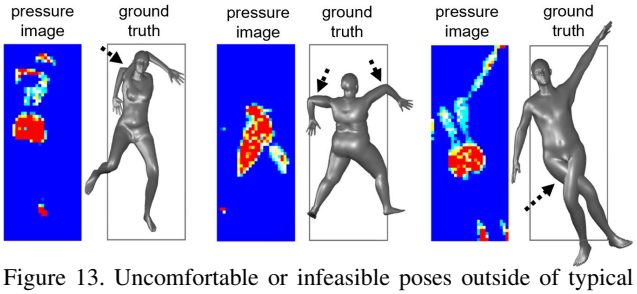

Figure 13. Uncomfortable or infeasible poses outside of typical human movement range (left, middle). Impossible pose where the thighs are in collision (right).

Appendix B: PressureNet

B.1. PressureNet Architecture Details

CNN - Convolutional Neural Network. Our CNN architecture, depicted in Fig. 14, is similar to that of Clever et al. [4], and uses the same kernel sizes, layers, and dropout. The first layer is a convolutional layer with a 7x7 kernel, and uses a stride of 2 and zero padding of size 3 on the sides of input images. The max pooling layer has a stride of 2 and padding of 0. All other convolutional layers are 3x3 with a stride of 1 and padding of 0. We use 192 channels in the first two convolutional layers and in the max pooling layer, and 384 channels in the last two convolutional layers. This CNN also differs from [4] in that we use tanh activa-

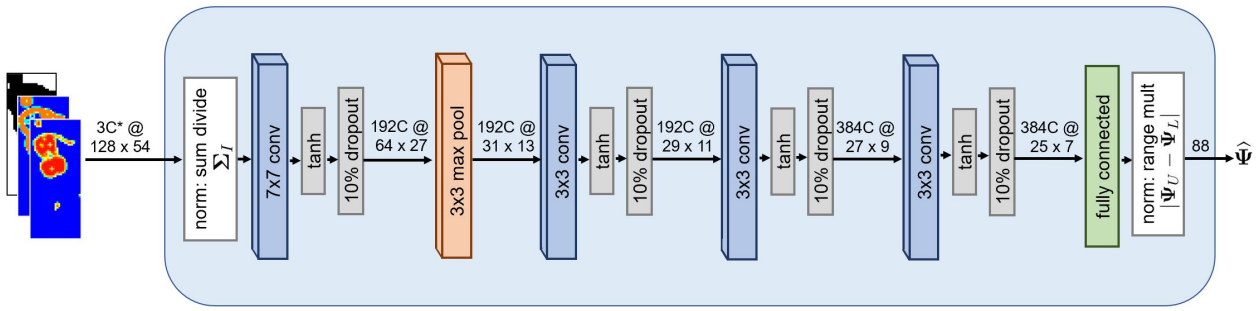

Figure 14. PressureNet: Convolutional Neural Network (CNN) with five convolutional layers, one max pooling layer, and one fully connected layer. Input images are normalized by per-image division by the sum of taxels. * indicates that the number of channels shown (3) represents Mod1 in Fig. 6 (a), whereas Mod2 in Fig. 6 (a) uses 5 input channels.

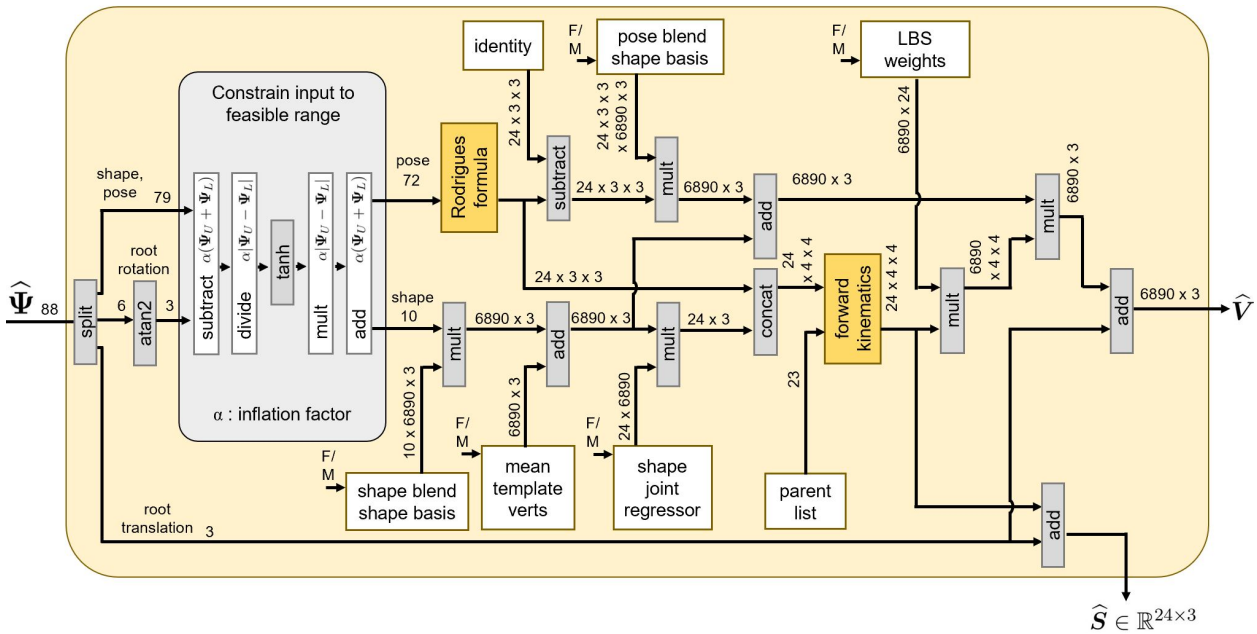

Figure 15. PressureNet: Differentiable SMPL human mesh reconstruction from Kanazawa et al. [6]. Our additions to [6] include input constraints (shown in the light grey box) and the root joint rotation and translation.

tion functions instead of ReLU. Through informal testing on smaller data sizes (e.g. 46K images), we observed that networks with tanh activations had less overfitting. We normalize the input and output of the network. To normalize the input channels, divide by the sum of taxels for each input image, $\mathbf{\Sigma}_I$. To normalize the output, we multiply it by the range of shape, pose, and posture parameters from the synthetic training dataset. We compute the range from the lower and upper limits, $\mathbf{\Psi}_L$ and $\mathbf{\Psi}_U$, of all parameters in the training dataset. For joint angle limits (i.e. pose), we use values from [12, 3, 11]. For body shape, we use sampling bounds $[-3, 3]$ from [10]. For global rotation, we use our sampling bounds for roll and yaw of $[-\pi, \pi]$ and $[-\frac{\pi}{6}, \frac{\pi}{6}]$, and for global translation, we use the size of the bed.

SMPL - Human Mesh Reconstruction. Following the CNN, we use the human model generative part of the HMR network [6], which inputs estimated shape, pose, and posture $\hat{\mathbf{\Psi}}$, and outputs a differentiable human mesh reconstruction $\hat{\boldsymbol{V}}$, as well as a set of $N = 24$ Cartesian joint positions $\hat{\boldsymbol{S}}$. This generative SMPL model, implemented in PyTorch [8], along with our modifications, is presented in Fig. 15.

In addition to using the generative kinematic SMPL embedding part of the full HMR network, our implementation constrains the input parameters to keep angles within human limits and body shape parameters inside our initial sampling range. To constrain the input parameters, we normalize the parameters to a range $[-1, 1]$ based on the limits $\mathbf{\Psi}_L, \mathbf{\Psi}_U$, and use a tanh function for a soft limit that is more amenable to gradient descent. Then, we perform a reverse normalization to scale back up. To prevent the tanh from clipping feasible values at the angle limits, for example a

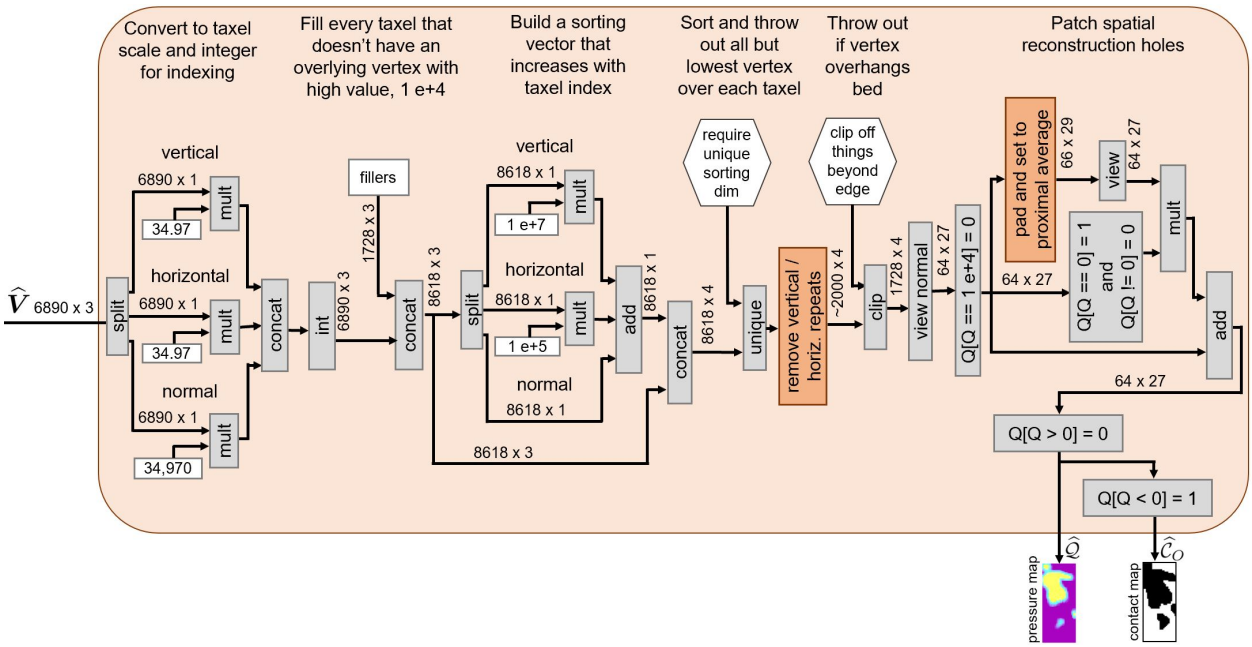

Figure 16. PressureNet: Pressure Map Reconstruction (PMR). PMR is fully differentiable, and performs sorting, filtering and patching to reconstruct spatial maps from the human mesh.

straight knee that is at 0 degrees, we inflate the angle range by a factor $\alpha = 1.2$ as shown in the figure.

PMR - Pressure Map Reconstruction. PMR, a novel component of PressureNet, takes as input a human mesh in global space $\hat{\boldsymbol{V}}$, and outputs a set of reconstructed spatial maps $\{\widehat{\mathcal{Q}}, \widehat{\mathcal{C}}_O\}$, which resemble a real pressure image and indicate where contact occurs between the estimated mesh and the bed. We reconstruct these maps differentiably as depicted in Fig. 16, meaning that we can backpropagate gradients through PMR to train the CNN. The PMR loss is based on the error between estimated spatial maps $\{\widehat{\mathcal{Q}}, \widehat{\mathcal{C}}_O\}$ and ground truth spatial maps $\{\mathcal{Q}, \mathcal{C}_O\}$. PMR works by projecting the mesh onto the surface of the bed and computing the distance that it sinks into the bed over each taxel. This amounts to finding the distance between the lowest vertex within the 2.9×2.9 cm area of each taxel and the undeformed height of the bed.

The PMR input $\hat{\boldsymbol{V}}$ is in units of meters, which we convert to units of taxels (1 m $\sim$ 35 taxels), so it can be indexed on the scale of the pressure image. We then use a process involving sorting, filtering, and patching to recreate the spatial maps, which is detailed in Fig. 16.

B.2. PressureNet Loss Function

We compute a loss on joint error rather than vertex error because the vertices are highly concentrated in some areas like the face and hands for aesthetic reasons, rather than for representing overall pose. Moreover, training the first network module ("Mod1") with reconstruction of 24 joint positions rather than a full set of vertices is much faster.

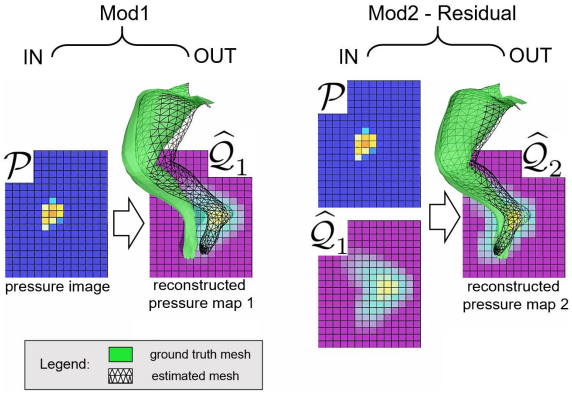

Figure 17. PressureNet deep learning in action, showing an example from our synthetic test set. The first network module ("Mod1") outputs an initial coarse pose estimate (right leg shown) and a reconstructed pressure map $\widehat{\mathcal{Q}}_1$. The second network module ("Mod2") corrects the estimated black mesh by a small angle difference based on the spatial residual between $\mathcal{P}$ and $\widehat{\mathcal{Q}}_1$.

The purpose of the second network module ("Mod2") is to fine-tune an initial estimate from Mod1 using both reconstructed pressure maps as input and a loss function with spatial map awareness. Fig. 17 shows a real example of how Mod2 corrects the initial mesh estimate from Mod1 using PMR. Note the spatial difference in the input images for Mod2, where the reconstructed map of the foot pressure in $\widehat{\mathcal{Q}}_1$ is shifted further right than the information on pressure image $\mathcal{P}$.

pose partition	test ct. real	test ct. synth	3DVPE real (cm)	3DVPE synth (cm)
supine straight limbs	76	1000	3.71	2.68
supine general	159	2000	4.51	3.40
supine crossed legs	78	1000	4.49	3.41
prone hands up	80	1000	5.12	4.24
general, roll $\sim \mathcal{U}[-\pi, \pi]$	479	6000	5.39	4.30
supine hands behind head	80	1000	5.09	4.40
gender partition				
F	480	6000	4.88	3.85
M	472	6000	5.10	4.04

Table 5. Partitioned results for prescribed poses with the best network for each real and synthetic.

B.3. PressureNet Training Details

We build PressureNet in PyTorch [9], which is shown at a high level in Figure 6 (b). For both Mod1 and Mod2, we used a learning rate of 0.00002 and a weight decay of 0.0005, which are the same used in [4]. We used the Adam optimizer for gradient descent [7]. Training Mod2 for 100 epochs using 184K images took 3 days on a Nvidia Tesla K80 GPU. Training Mod2 took 8 days due to increased computation from PMR.

B.4. Results for Separate Partitions

Table 5 shows the results of our PressureNet evaluated between prescribed resting poses from participants in bed, and a per-gender comparison.

B.5. Additional Failure Cases

We present additional failure cases in Fig. 18. One limitation is that our network does not have an interpenetration error, so the limbs sometimes intersect, e.g. the left hand in Fig. 18(a)-top left. Our network also failed for some limbs when there was little or no contact information, and for non-resting poses. This issue is related to the limitations of the sensor, which were explored in [4]. Our network failed for non-resting poses, such those in [4]; however these are not part of the training or testing PressurePose dataset. We observed some inaccuracies when testing on training data (Figs. 9 and 18), which suggests that there is a performance limitation on the network's ability to extract pressure image features in some scenarios.

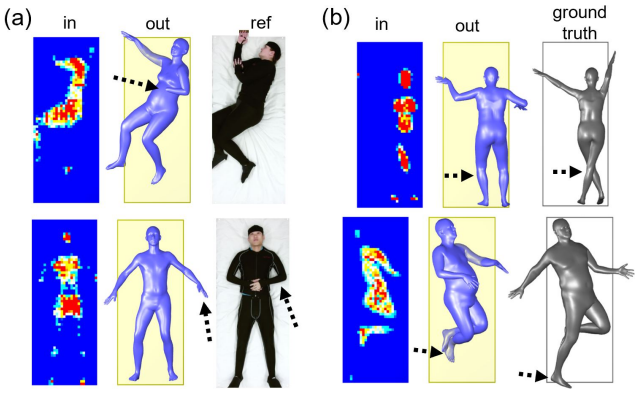

Figure 18. (a) Real data failure cases. Self penetration of inferred left hand into chest (top), lack of information on mat leading to inaccurate pose (bottom). (b) Synthetic data failure cases: testing on *training* data, various inaccuracies.

References

[1] I. Akhter and M. J. Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *CVPR*, 2015. 5

[2] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *ECCV*, pages 561–578. Springer, 2016. 2

[3] Donna C. Boone and Stanley P. Azen. Normal range of motion of joints in male subjects. *Journal of Bone and Joint Surgery*, 61(5):756–759, 1979. 6

[4] Henry M. Clever, Ariel Kapusta, Daehyung Park, Zackory Erickson, Yash Chitalia, and Charles C. Kemp. 3D human pose estimation on a configurable bed from a pressure image. In *IROS*, pages 54–61. IEEE, 2018. 2, 5, 8

[5] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003. 3, 4

[6] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, pages 7122 – 7131. IEEE, 2018. 6

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 8

[8] MandyMo. PyTorch HMR - https://github.com/MandyMo/pytorch_HMR, 2018. 6

[9] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 8

[10] Anurag Ranjan, Javier Romero, and Michael J. Black. Learning human optical flow. In *British Machine Vision Conference*. BMVA Press, 2018. 6

[11] Asbjørn Roaas and Gunnar BJ Andersson. Normal range of motion of the hip, knee and ankle joints in male subjects, 3040 years of age. *Acta Orthopaedica Scandinavica*, 53(2):205–208, 1982. 6

[12] J. M. Soucie, C. Wang, A. Forsyth, S. Funk, M. Denny, K. E. Roach, D. Boone, and Hemophilia Treatment Center Network. Range of motion measurements: reference values and a database for comparison studies. *Haemophilia*, 17(3):500–507, 2011. 6

[13] Aydin Tözeren. *Human Body Dynamics: Classical Mechanics and Human Movement*. Springer, 1999. 2

[14] Thiemo Wiedemeyer. IAI Kinect2. https://github.com/code-iai/iai_kinect2, 2014 – 2015. Accessed June 12, 2015. 5