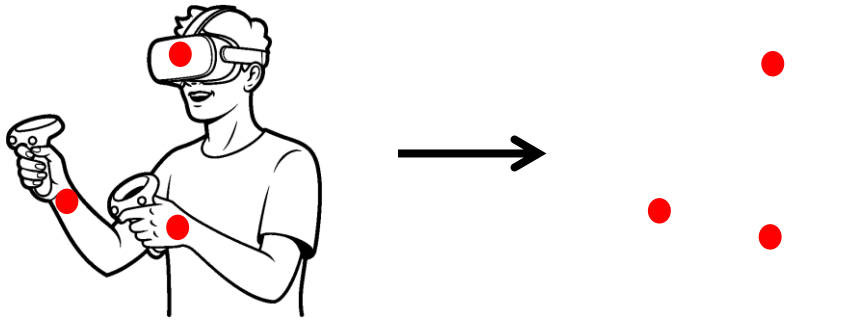
Zero-shot Human Pose Estimation using Diffusion-based Inverse solvers

Sahil Bhandary Karnoor, Romit Roy Choudhury

University of Illinois Urbana-Champaign

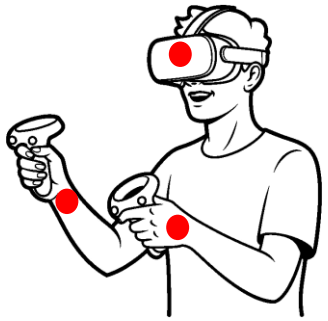
Problem Statement



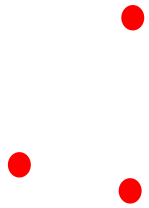
Real-world use case

Sparse motion measurement

Problem Statement



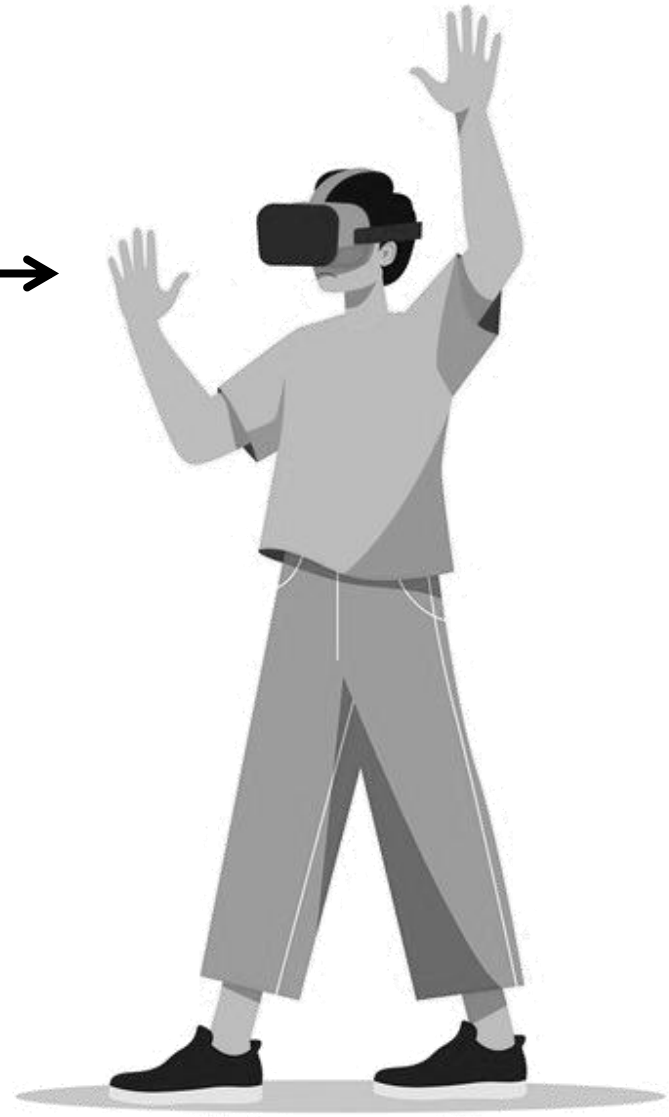
Real-world
use case



Sparse motion
measurement

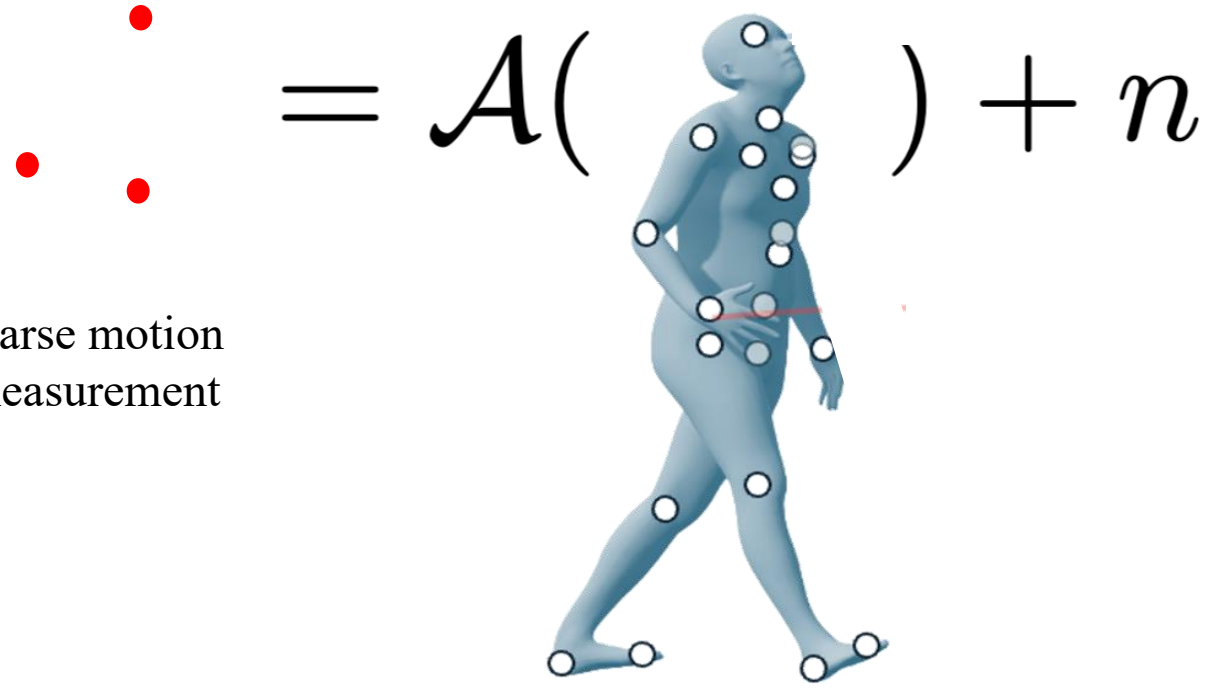
InPose
(our algorithm)

Full body
3D pose



Problem Statement

Formulate as an inverse problem

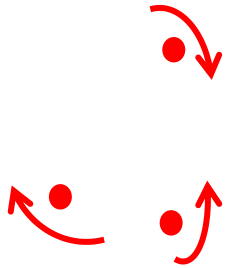


Sparse motion measurement

Full body 3D pose

Problem Statement

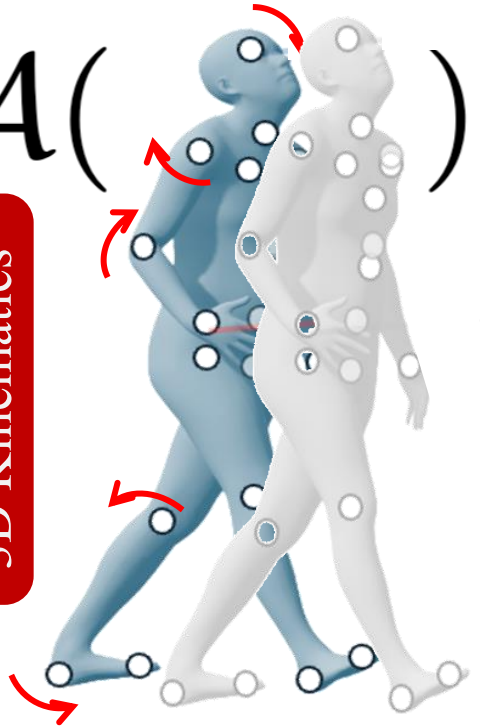
Formulate as an inverse problem



Sparse motion measurement

$$= \mathcal{A}(\quad) + n$$

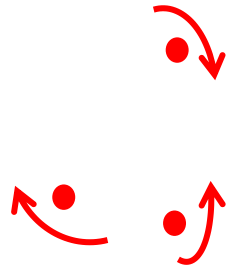
3D Kinematics



Full body 3D pose

Problem Statement

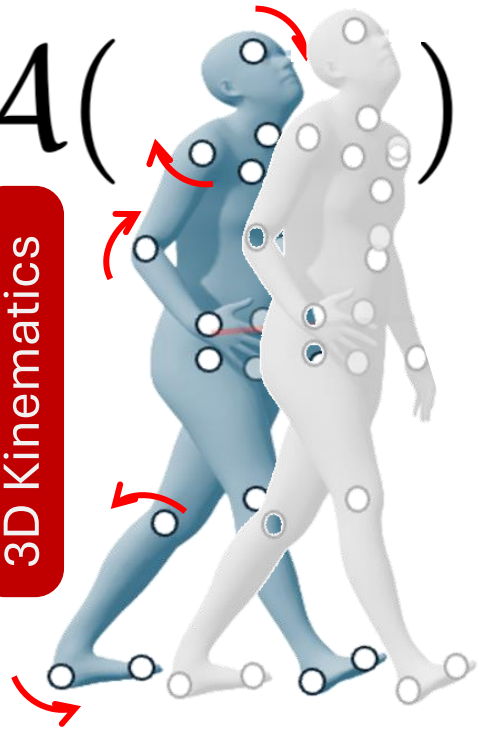
Formulate as an inverse problem



Sparse motion measurement

$$= \mathcal{A}(\quad) + n$$

3D Kinematics

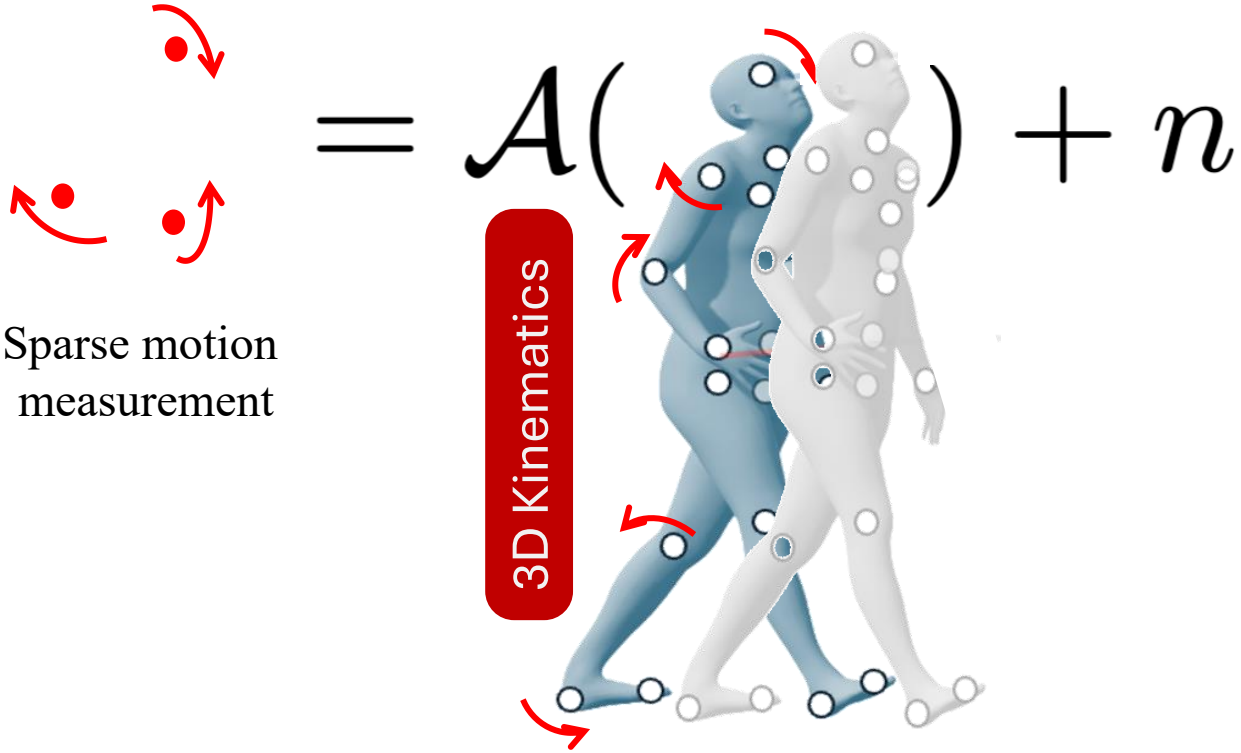


Full body 3D pose

We will solve this inverse problem using generative priors (from diffusion models)

Problem Statement

Formulate as an inverse problem

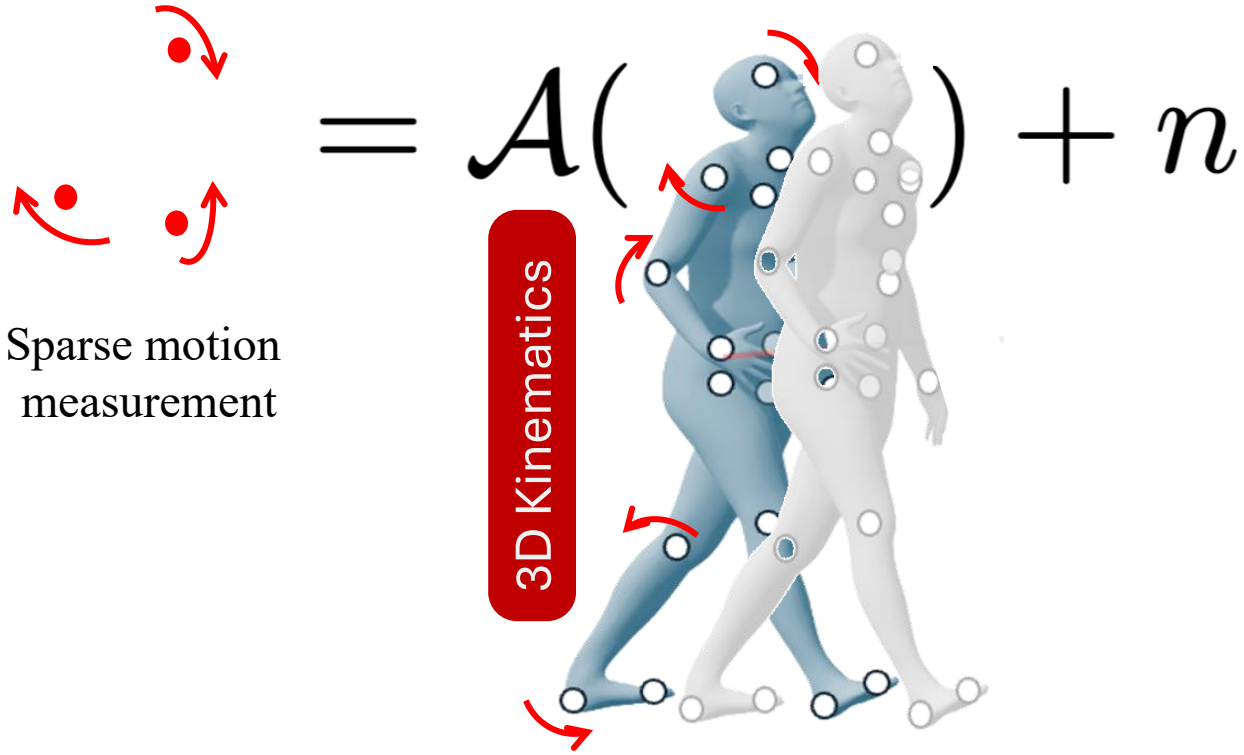


We will solve this inverse problem using generative priors (from diffusion models)

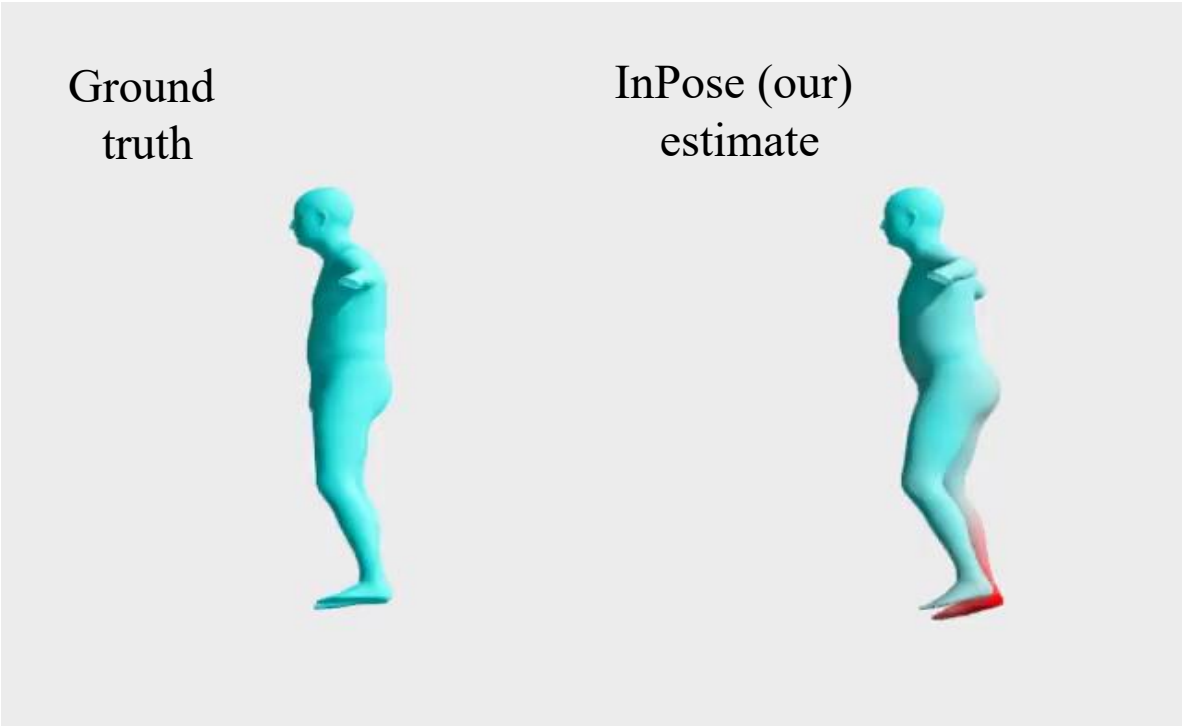
Generalizable to any body shape (with no user specific training)

Problem Statement

Formulate as an inverse problem



… and show results as follows:



We will solve this inverse problem using generative priors (from diffusion models)

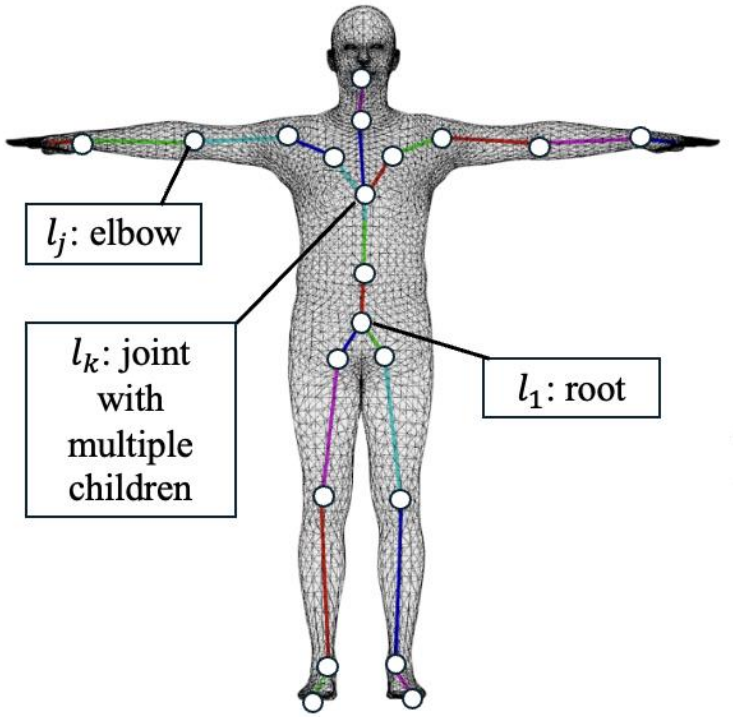
Generalizable to any body shape (with no user specific training)

Roadmap

- Problem formulation
  - Kinematic tree
  - Scale dependent vs. scale free pose
  - 3-point pose estimation
- Learning representations
  - Rotation representation
  - 6D representation
- Background on Diffusion
  - Score function
  - Guided diffusion and Posterior sampling
  - Pseudo-inverse based posterior sampling (ΠGDM)
- InPose: Our work
  - Limitations of past work
  - Key intuition: learning scale free parameters
  - Diffusion priors and ΠGDM linear model
  - Theorem
  - InPose pipeline
  - Results

Pose Estimation

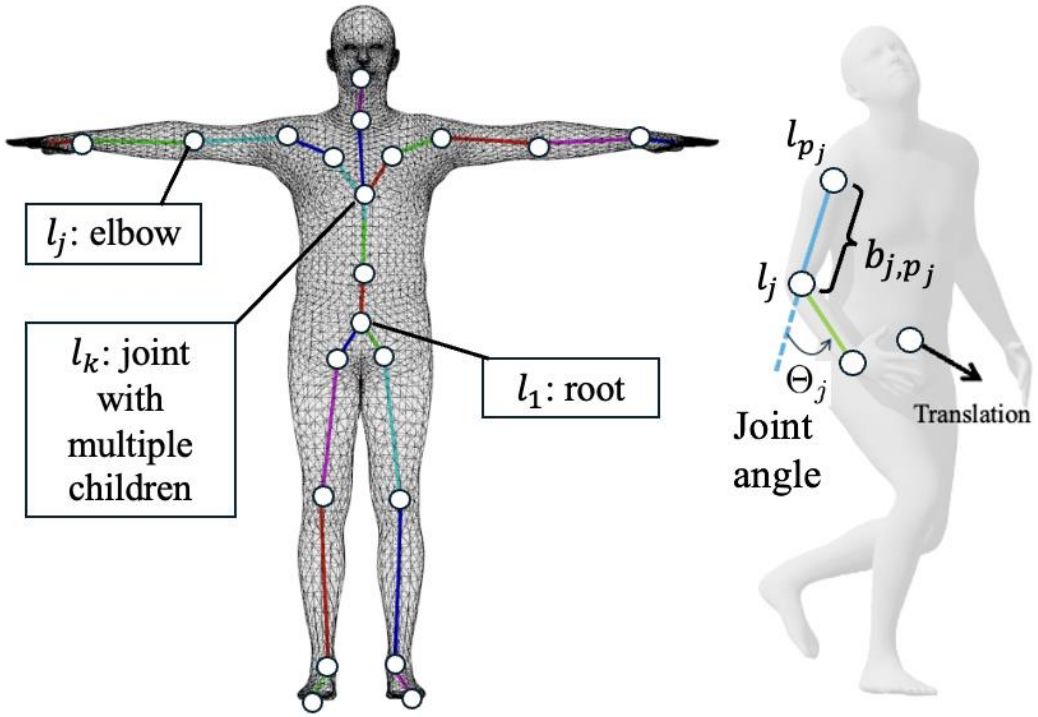
- Pose refers to the state of all joints l_j, $j \in M$ typically, 22 in number.



Pose Estimation

- Pose refers to the state of all joints l_j, $j \in M$ typically, 22 in number.
- Joints have angle $\Theta_j \in SO(3)$ and bone $b_{j,p_j} \in \mathbb{R}^3$ attached to parent p_j.
- Joint states are thus coordinate transforms:

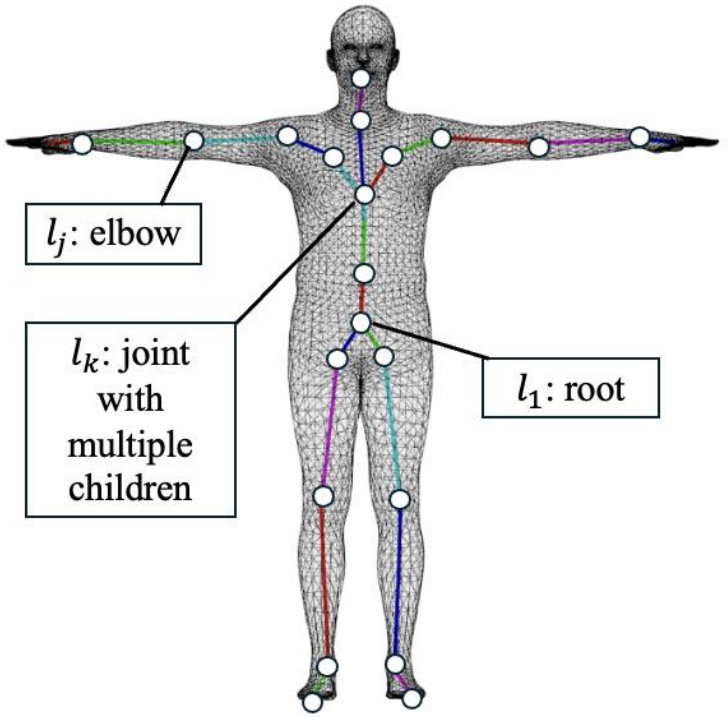
$$ {}^{p_j}T_j = \begin{bmatrix} \Theta_j & b_{j,p_j} \\ 0^{1\times 3} & 1 \end{bmatrix} $$

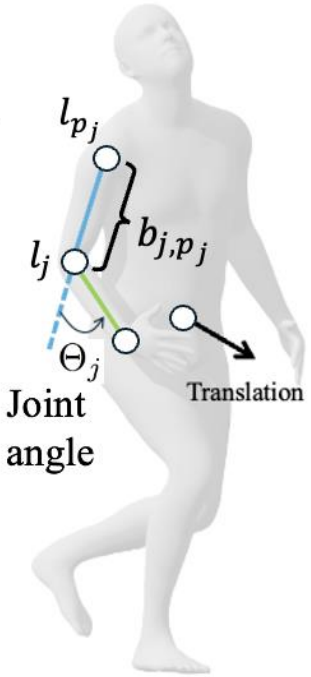


Pose Estimation

- Pose refers to the state of all joints l_j, $j \in M$ typically, 22 in number.
- Joints have angle $\Theta_j \in SO(3)$ and bone $b_{j,p_j} \in \mathbb{R}^3$ attached to parent p_j.
- Joint states are thus coordinate transforms:

$$ {}^{p_j}T_j = \begin{bmatrix} \Theta_j & b_{j,p_j} \\ 0^{1\times 3} & 1 \end{bmatrix} $$

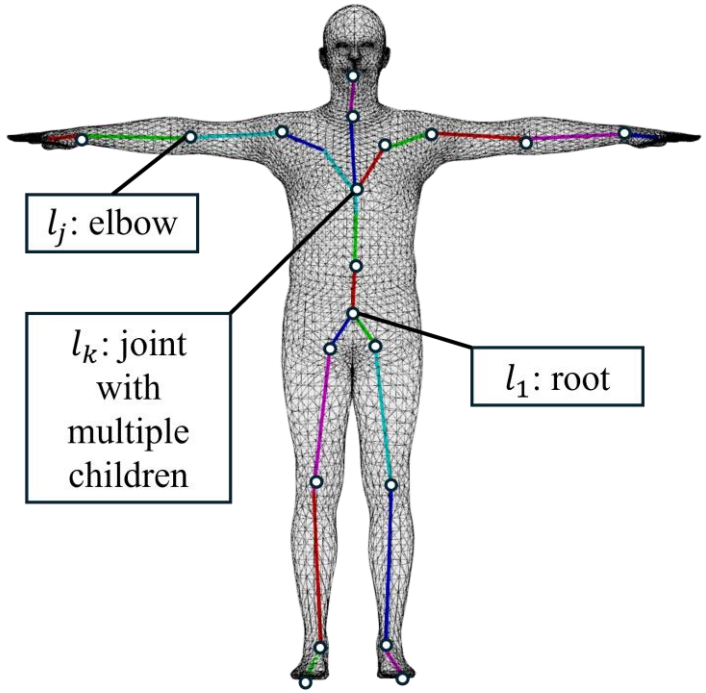




People with different body shapes have different bone lengths for b_{j,p_j}

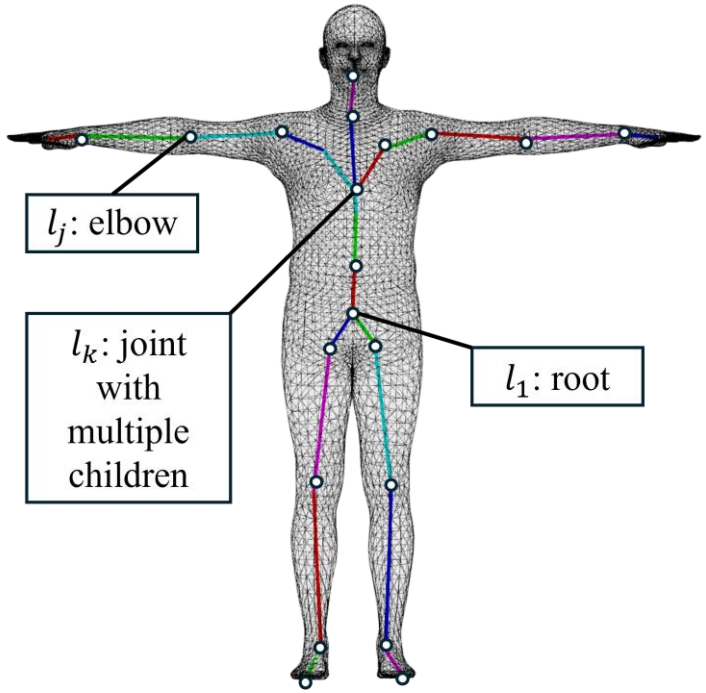
Kinematic Tree

- Joints define a directed acyclic graph - joints as vertices, parent-joint pairs as edges, with a single parent for each joint.



Kinematic Tree

- Joints define a directed acyclic graph - joints as vertices, parent-joint pairs as edges, with a single parent for each joint.
- Thus, we can define a kinematic tree with the pelvis at the root (with state ${}^{G}T_1$).



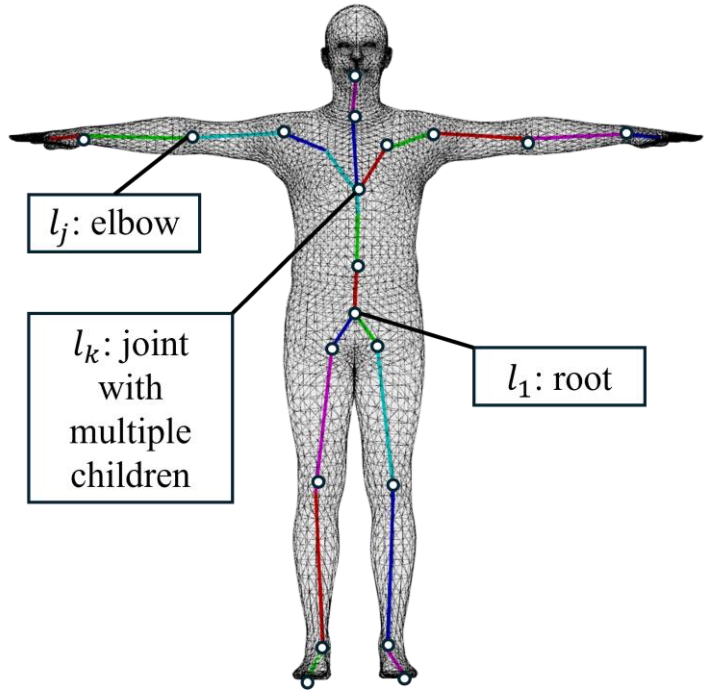
Kinematic Tree

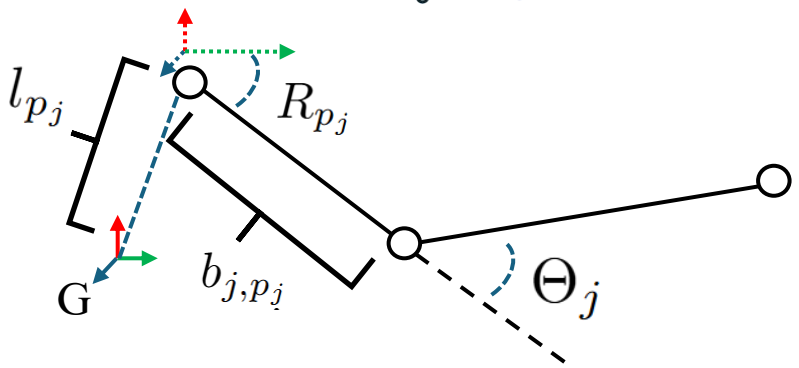
- Joints define a directed acyclic graph - joints as vertices, parent-joint pairs as edges, with a single parent for each joint.
- Thus, we can define a kinematic tree with the pelvis at the root (with state ${}^{G}T_1$).

- Useful to recursively calculate joint state in global frame:

$$ {}^{G}T_j = {}^{G}T_{p_j} \cdot {}^{p_j}T_j, \qquad {}^{G}T_j = \begin{bmatrix} R_j & l_j \\ 0^{1\times 3} & 1 \end{bmatrix} $$

- We term $l_j \in \mathbb{R}^3$ joint location and $R_j \in SO(3)$ the global joint rotation.



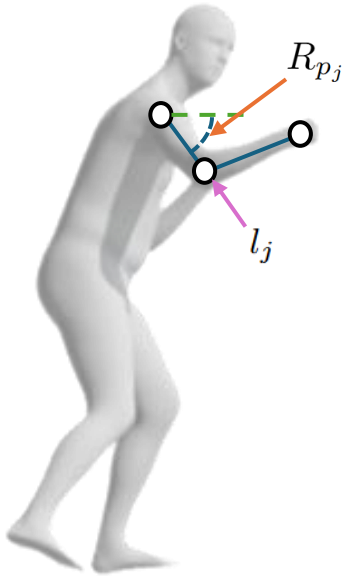


Scale-dependent vs Scale-free Pose

- Using the tree, we can recursively track joint locations and global joint rotations:

$$R_j = R_{p_j} \cdot \Theta_j \qquad (1a)$$

$$l_j = l_{p_j} + R_{p_j} \cdot b_{j,p_j} \qquad (1b)$$



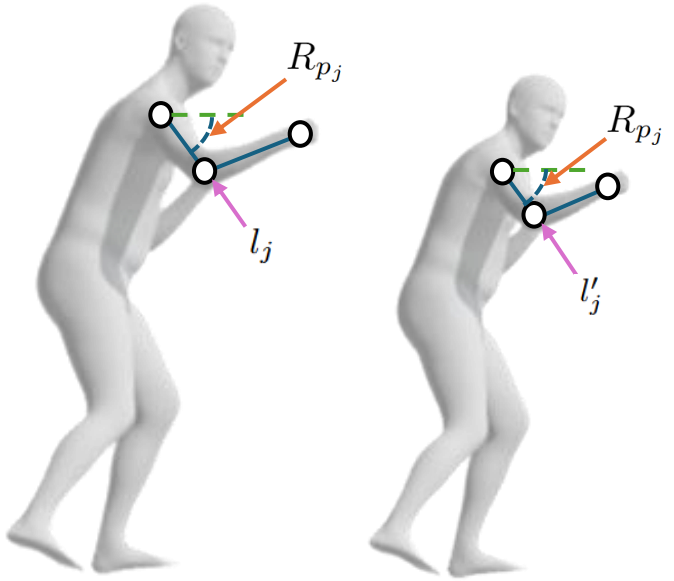
Scale-dependent vs Scale-free Pose

- Using the tree, we can recursively track joint locations and global joint rotations:

$$R_j = R_{p_j} \cdot \Theta_j \qquad (1a)$$

$$l_j = l_{p_j} + R_{p_j} \cdot b_{j,p_j} \qquad (1b)$$

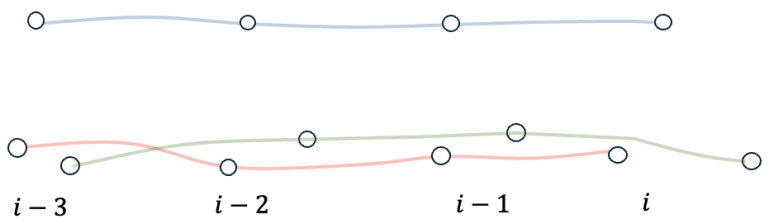
- Users in the same "Pose" have same joint angles regardless of body shape (or bones b_{j,p_j}).
- We thus call joint angles $\{R_j\}$ Scale-free pose and joint locations $\{l_j\}$ Scale-dependent pose.



3-Point Pose Estimation

- AR/VR – Estimate Full Body Pose from underdetermined sensing.
- Observations - $y_m(i) = [l_m(i), R_m(i)]$, where

  m := {head, left wrist, right wrist} joints.



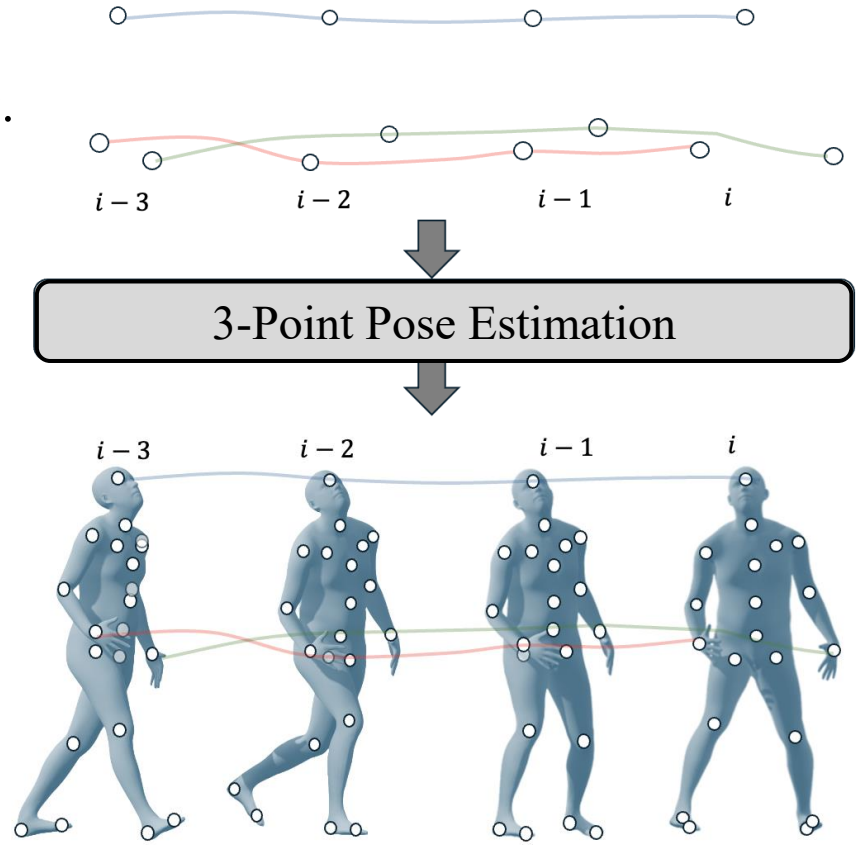
3-Point Pose Estimation

- AR/VR – Estimate Full Body Pose from underdetermined sensing.
- Observations - $y_m(i) = [l_m(i), R_m(i)]$, where

  m := {head, left wrist, right wrist} joints.

- Estimate $\{l_M(i), R_M(i)\}$ from $y_m(i)$

- Measurements are assumed to be contaminated by Gaussian Noise: $l_m(i) = l_m^+(i) + \sigma_l v(i)$



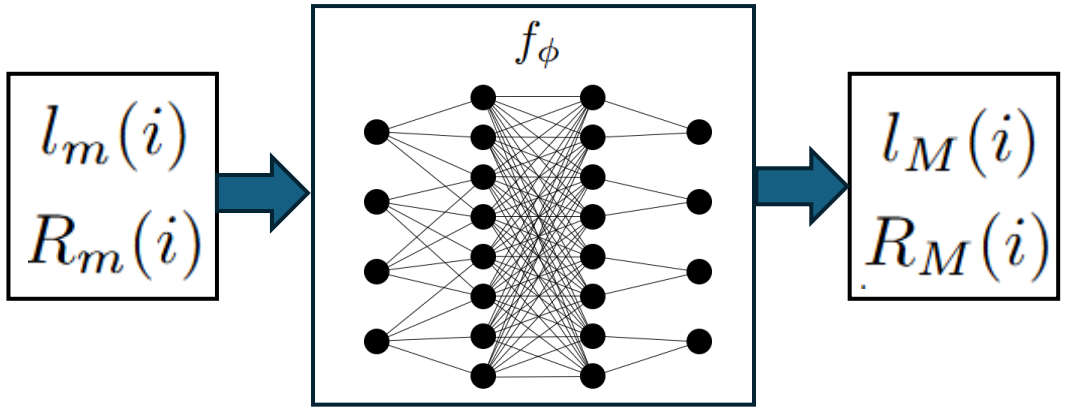
Roadmap

- Problem formulation
  - Kinematic tree
  - Scale dependent vs. scale free pose
  - 3-point pose estimation
- Learning representations
  - Rotation representation
  - 6D representation
- Background on Diffusion
  - Score function
  - Guided diffusion and Posterior sampling
  - Pseudo-inverse based posterior sampling (ΠGDM)
- InPose: Our work
  - Limitations of past work
  - Key intuition: learning scale free parameters
  - Diffusion priors and ΠGDM linear model
  - Theorem
  - InPose pipeline
  - Results

Deep Learning for 3-Point Pose Estimation

- Learn a Neural Network f_ϕ to predict pose:

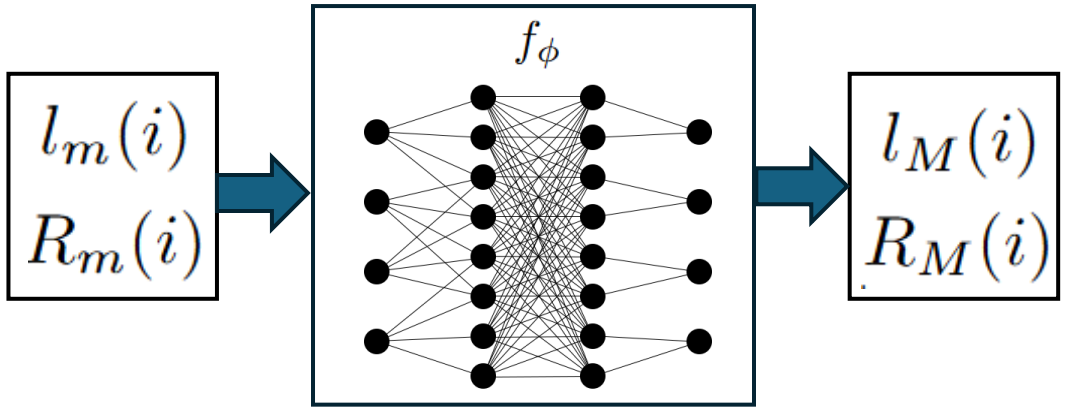
$$\{l_M(i), R_M(i)\} = f_\phi(l_m(i), R_m(i))$$



Deep Learning for 3-Point Pose Estimation

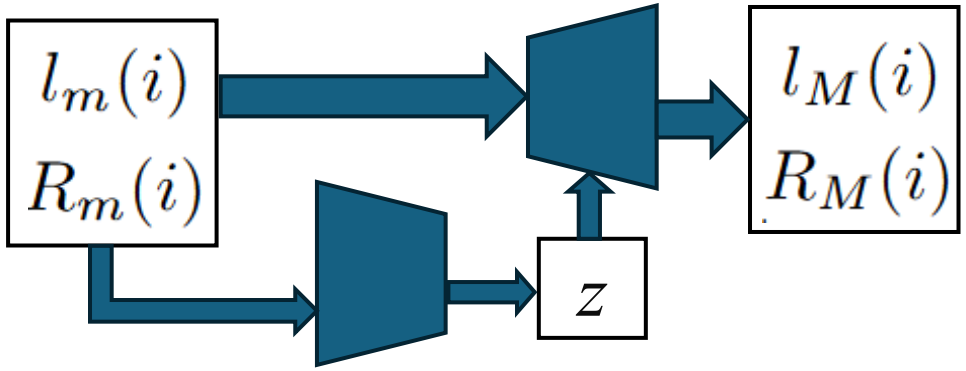
- Learn a Neural Network f_ϕ to predict pose:

$$\{l_M(i), R_M(i)\} = f_\phi(l_m(i), R_m(i))$$



- Use a Conditional VAE to learn the Posterior:

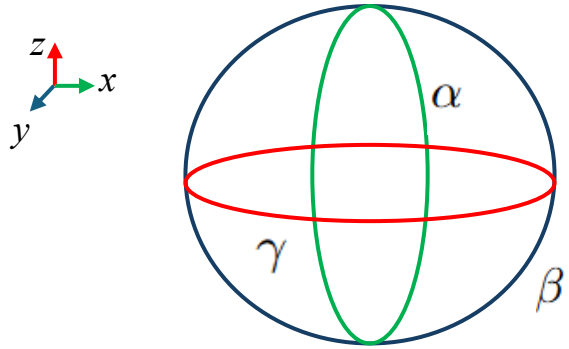
$$\underset{l_M(i), R_M(i)}{\arg\max}\ p(l_M(i), R_M(i) | l_m(i), R_m(i))$$



- Guided Diffusion, etc.

Rotation Representation for Deep Learning

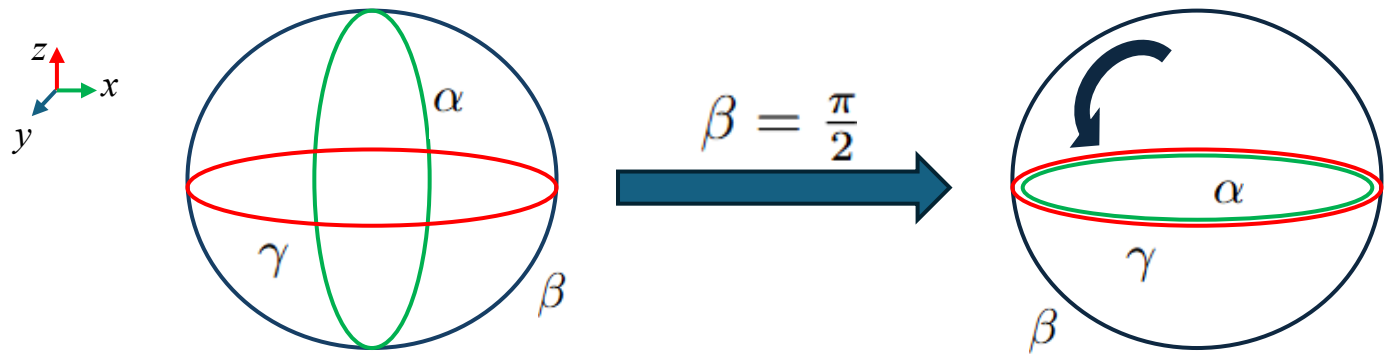
- Options To represent Rotations – Euler, Quaternion, Rotation Matrix
- Issues with various representations:
  - Euler angles



Rotation Representation for Deep Learning

- Options To represent Rotations – Euler, Quaternion, Rotation Matrix
- Issues with various representations:
  - Euler angles – Gimbal lock
  - E.g. at $\alpha, \beta = \frac{\pi}{2}, \gamma$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}$$



Rotation Representation for Deep Learning

- Issues with various representations:
  - Quaternion –

$$q = [w \quad ix \quad jy \quad kz]^\top$$

$$q(R) = \begin{bmatrix} \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \\ i\ \text{sgn}(R_{32} - R_{23})\left|\frac{1}{2}\sqrt{1 + R_{31} + R_{22} + R_{33}}\right| \\ j\ \text{sgn}(R_{13} - R_{31})\left|\frac{1}{2}\sqrt{1 - R_{31} + R_{22} - R_{33}}\right| \\ k\ \text{sgn}(R_{21} - R_{12})\left|\frac{1}{2}\sqrt{1 - R_{31} - R_{22} + R_{33}}\right| \end{bmatrix}$$

Rotation Representation for Deep Learning

- Issues with various representations:
  - Quaternion – Double Cover

    (Discontinuous representation)

$$q = [w \quad ix \quad jy \quad kz]^\top$$

$$q(R) = \begin{bmatrix} \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \\ i\ \mathrm{sgn}(R_{32} - R_{23})\left|\frac{1}{2}\sqrt{1 + R_{31} + R_{22} + R_{33}}\right| \\ j\ \mathrm{sgn}(R_{13} - R_{31})\left|\frac{1}{2}\sqrt{1 - R_{31} + R_{22} - R_{33}}\right| \\ k\ \mathrm{sgn}(R_{21} - R_{12})\left|\frac{1}{2}\sqrt{1 - R_{31} - R_{22} + R_{33}}\right| \end{bmatrix}$$

  - E.g. Rotation by $\gamma = \pi$

$$R(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$q(R(\pi-)) = [0 \quad 0 \quad 0 \quad 1]^\top$$

$$q(R(\pi+)) = [0 \quad 0 \quad 0 \quad -1]^\top$$

6D Rotation Representation

- Issues with various representations:
  - Rotation matrices – a predicted matrix

    $R = [c^1\ c^2\ c^3]$ may not necessarily have orthonormal

    columns, e.g. $c^1 \times c^2 \neq c^3$

- Columns need to be postprocessed.

6D Rotation Representation

- Issues with various representations:
  - Rotation matrices – a predicted matrix

    $R = [c^1\ c^2\ c^3]$ may not necessarily have orthonormal columns, e.g. $c^1 \times c^2 \neq c^3$

- Columns need to be postprocessed.

- Directly use the first 2 columns (6D)[1]:

$$r(R) = [c^{1\top}, c^{2\top}]^\top \in \mathbb{R}^6$$

[1] Yi et al. "On the continuity of rotation representations in neural networks." (CVPR), June 2019

6D Rotation Representation

- Issues with various representations:
  - Rotation matrices – a predicted matrix $R = [c^1\ c^2\ c^3]$ may not necessarily have orthonormal columns, e.g. $c^1 \times c^2 \neq c^3$
- Columns need to be postprocessed.
- Directly use the first 2 columns (6D)[1]:

$$r(R) = [c^{1\top}, c^{2\top}]^\top \in \mathbb{R}^6$$

- Then, use the non-linear function $\mathcal{D}()$ to recover $R_j(i)$
- Most SoTA works use this representation

$$R_j(i) = \mathcal{D}(r_j(i))$$

$$c^1 = \left[r_j^{1:3}(t)\right]^\top,$$

$$c^2 = \left[r_j^{4:6}(t)\right]^\top - \text{proj}_{\bar{c}^1}\left(\left[r_j^{4:6}(t)\right]^\top\right),$$

$$\bar{c}^1 = \frac{c^1}{||c^1||} \quad \bar{c}^2 = \frac{c^2}{||c^2||} \quad \bar{c}^3 = \bar{c}^1 \times \bar{c}^2$$

$$R_j(t) = \left[\bar{c}^1 \quad \bar{c}^2 \quad \bar{c}^3\right]$$

[1] Yi et al. "On the continuity of rotation representations in neural networks." (CVPR), June 2019

Roadmap

- Problem formulation
  - Kinematic tree
  - Scale dependent vs. scale free pose
  - 3-point pose estimation
- Learning representations
  - Rotation representation
  - 6D representation
- Background on Diffusion
  - Score function
  - Guided diffusion and Posterior sampling
  - Pseudo-inverse based posterior sampling (ΠGDM)
- InPose: Our work
  - Limitations of past work
  - Key intuition: learning scale free parameters
  - Diffusion priors and ΠGDM linear model
  - Theorem
  - InPose pipeline
  - Results

Diffusion Models

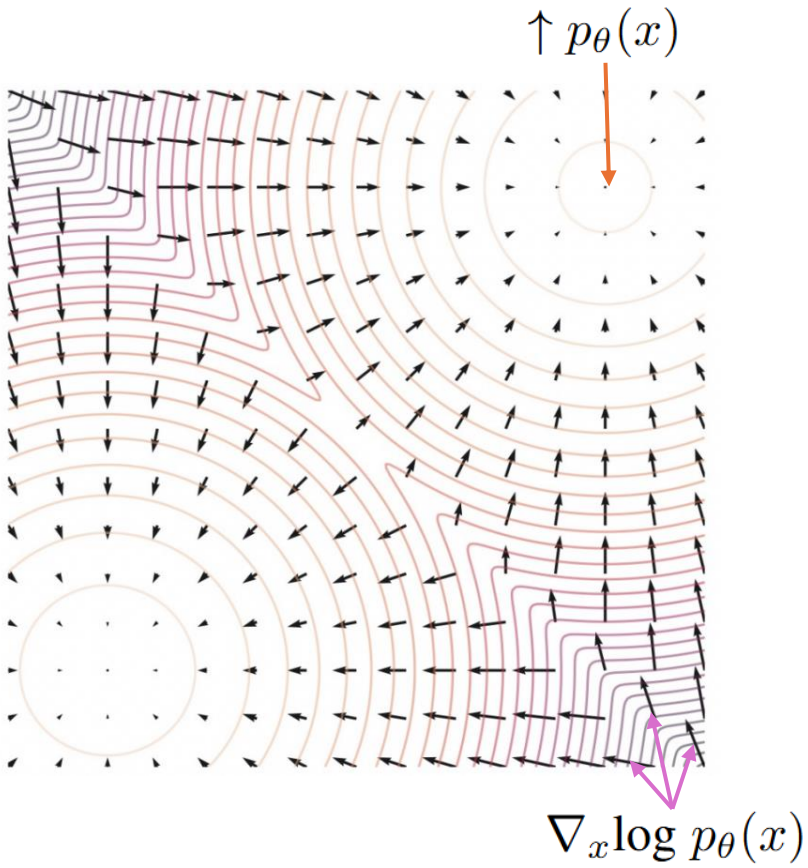
- Let's say we would like to learn a prior $p_\theta(x)$ for some dataset, with $x \in \mathbb{R}^D$
- Can be used to sample from this distribution

Diffusion Models

- Let's say we would like to learn a prior $p_\theta(x)$ for some dataset, with $x \in \mathbb{R}^D$

- We could learn the gradient field $\nabla_x \log p_\theta(x)$
- Points in the direction of higher probability.

We could start at some random point and travel along the gradient field till we reach a sample with high probability



Diffusion Models

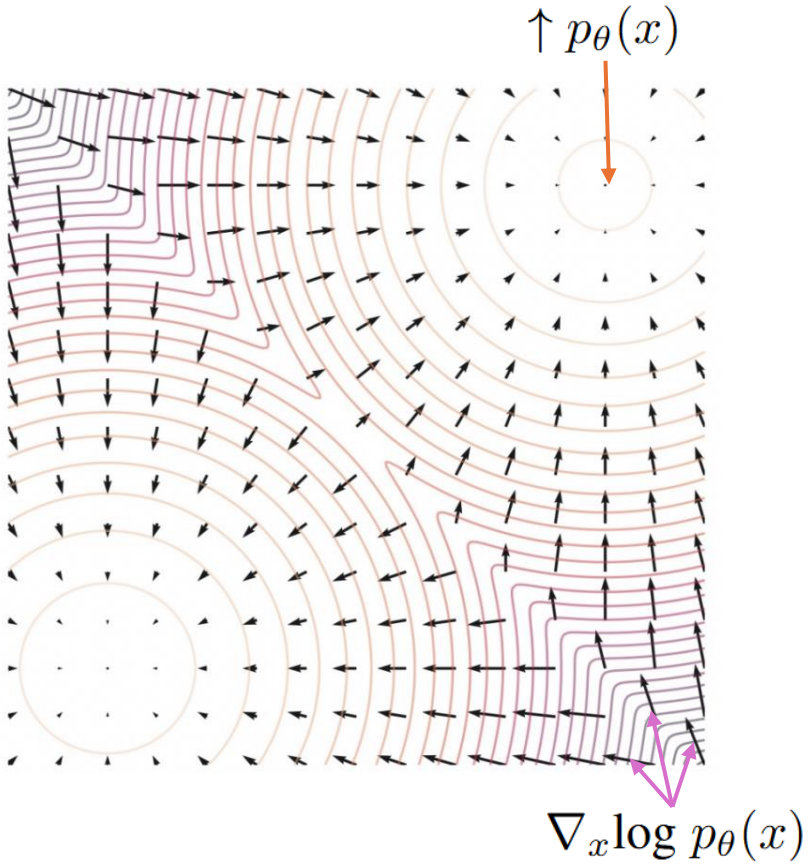
- Let's say we would like to learn a prior $p_\theta(x)$ for some dataset, with $x \in \mathbb{R}^D$

- We could learn the gradient field $\nabla_x \log p_\theta(x)$
- Points in the direction of higher probability.

- e.g. Stochastic Gradient Langevin Dynamics:
  - Sample from some easy to sample prior $x_T \sim \pi(x)$
  - Iterate with $z_t \sim \mathcal{N}(0, \mathrm{I}^D)$:

$$x_{t-1} = x_t + \epsilon \nabla_x \log p_\theta(x) + \sqrt{2\epsilon} z_t$$

  - As $\epsilon \to 0,\ T \to \infty$ we find $x_0 \sim p_\theta(x)$



How do we learn the gradient field $\nabla_x \log p_\theta(x)$, which is called the Score function

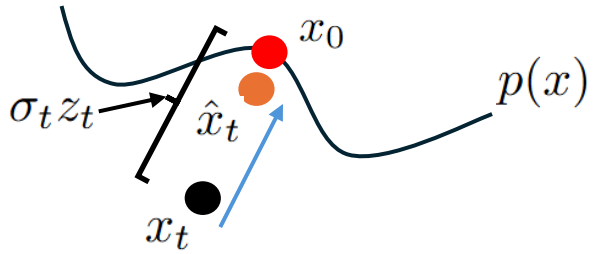
Score Function

- Let's say we noise a sample x_0 with $z_t \sim \mathcal{N}(0, \mathrm{I}^D)$

$$x_t = x_0 + \sigma_t z_t$$

- The MMSE estimate of x_0 is $\hat{x}_t = \mathbb{E}[x_0|x_t]$

- The MMSE can be thought of as a denoiser.

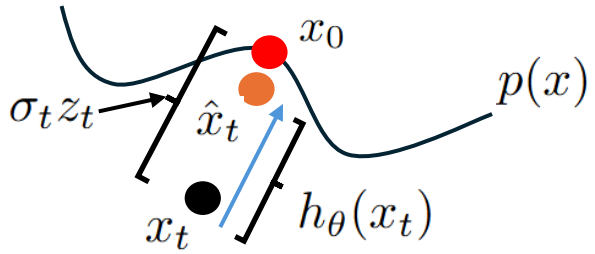


We have a rich body of literature studying denoisers

Score Function

- Let's say we noise a sample x_0 with $z_t \sim \mathcal{N}(0, \mathrm{I}^D)$

$$x_t = x_0 + \sigma_t z_t$$



- The MMSE estimate of x_0 is $\hat{x}_t = \mathbb{E}[x_0|x_t]$
- The MMSE can be thought of as a denoiser.
- We have a rich body of literature studying denoisers.
- Let's learn a Denoiser $h_\theta(x_t)$, using a technique called Denoising score matching(DSM):

$$L_{\text{DSM}}(\Theta) = \mathbb{E}_{x_0, x_t}[||h_\theta(x_t) - x_0||^2]$$

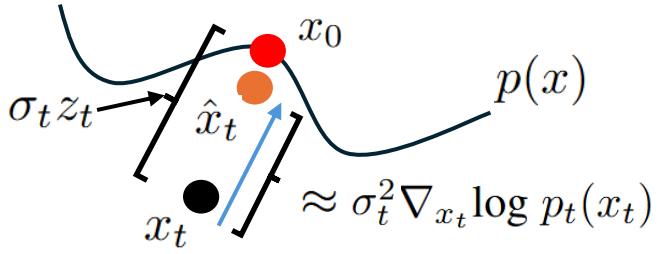
Score Function

- Let's say we noise a sample x_0 with $z_t \sim \mathcal{N}(0, \mathrm{I}^D)$

$$x_t = x_0 + \sigma_t z_t$$

- The MMSE estimate of x_0 is $\hat{x}_t = \mathbb{E}[x_0|x_t]$
- The MMSE can be thought of as a denoiser.
- Let's learn a Denoiser $h_\theta(x_t)$ using a technique called Denoising score matching(DSM):

$$L_{\text{DSM}}(\Theta) = \mathbb{E}_{x_0, x_t}[||h_\theta(x_t) - x_0||^2]$$



- There exists an important statistical result, called Tweedie's formula that states: $\nabla_{x_t} \log p_t(x_t) = \dfrac{\mathbb{E}[x_0|x_t] - x_t}{\sigma_t^2}$

Use $h_\theta(x_t)$ to approximate the score function:

$$\nabla_{x_t} \log p_t(x_t) \approx \frac{h_\theta(x_t) - x_t}{\sigma_t^2}$$

Algorithms for Diffusion

- Class of Generative Modeling algorithms that exploit the score function.

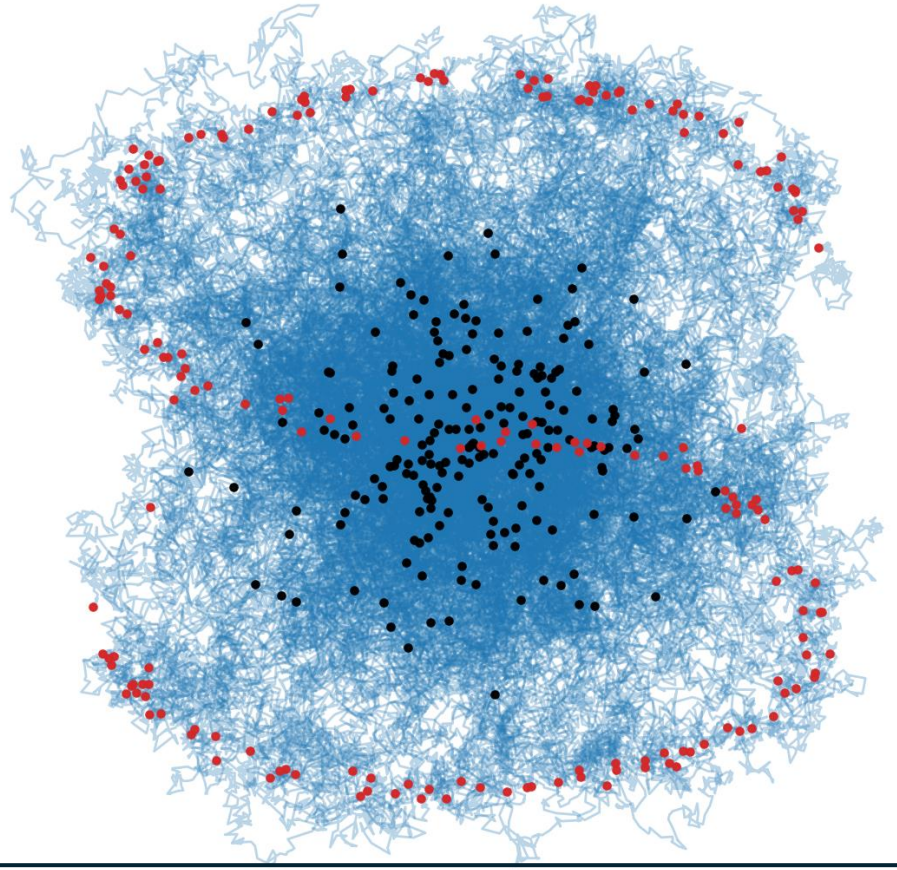
- Some important Diffusion algorithms:
  - Stochastic Gradient Langevin Dynamics.
  - Denoising Diffusion Probabilistic Models (DDPM).

DDPM (Stochastic)

$\sim p_\theta(x)$

$\sim \pi(x)$

Sample Path



Algorithms for Diffusion

- Class of Generative Modeling algorithms that exploit the score function.

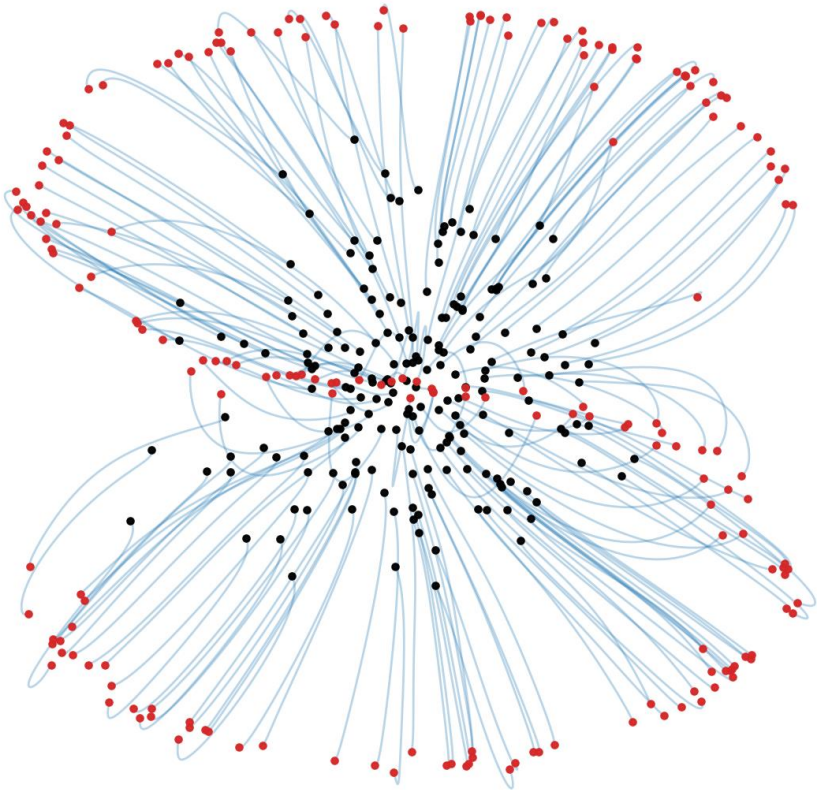
- Some important Diffusion algorithms:
  - Stochastic Gradient Langevin Dynamics.
  - Denoising Diffusion Probabilistic Models (DDPM).
  - Denoising Diffusion Implicit Models (DDIM).

DDIM (Deterministic)

$\sim p_\theta(x)$

$\sim \pi(x)$

Sample Path



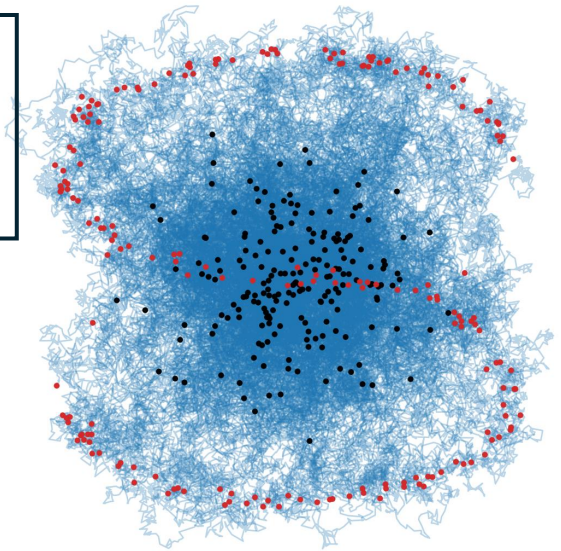
Algorithms for Diffusion

- Class of Generative Modeling algorithms that exploit the score function.

- Some important Diffusion algorithms:
  - Stochastic Gradient Langevin Dynamics.
  - Denoising Diffusion Probabilistic Models (DDPM).
  - Denoising Diffusion Implicit Models (DDIM).
  - Second order ODE solvers such as the Heun solver.

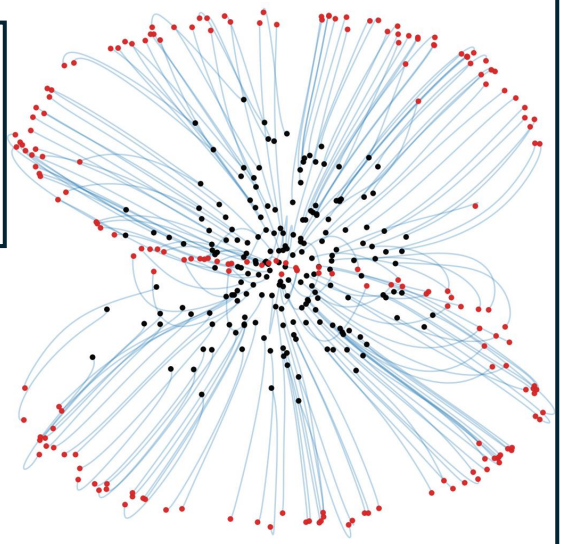
DDPM (Stochastic)

$\sim p_\theta(x)$
$\sim \pi(x)$
Sample Path



DDIM (Deterministic)

$\sim p_\theta(x)$
$\sim \pi(x)$
Sample Path



Guided Diffusion

- Let's say we are given a linear measurement $y = Ax_0 + n$, where $n \sim \sigma_y \mathcal{N}(0, \mathrm{I})$.
- We want samples $x_0 \sim p_\theta(x)$ that also belong to a Data distribution.
- In other words - sample from the posterior $p(x|y)$.

Can we use the Diffusion framework itself?

- Option 1: Train a conditional denoiser for Diffusion
  - Called Classifier-free Guidance (CFG):

Posterior Score

- Let's say we are given a linear measurement $y = Ax_0 + n$

- Option 2: Use the posterior score in the Diffusion framework:

$$\nabla_{x_t} \log p_t(x_t|y)$$

- Posterior Score – Substitute the prior score with $\nabla_{x_t} \log p_t(x_t|y)$

Can we derive $\nabla_{x_t} \log p_t(x_t|y)$ from the prior score without any training?

Posterior Score

- Option 2: Use the posterior score in the Diffusion framework: $\nabla_{x_t} \log p_t(x_t|y)$

- Using Bayes Rule, we can split the posterior score:

$$\nabla_{x_t} \log p_t(x_t|y) = \underbrace{\nabla_{x_t} \log p_t(x_t)}_{\text{Prior Score}} + \underbrace{\nabla_{x_t} \log p_t(y|x_t)}_{\text{Likelihood Score}}$$

- To estimate the Likelihood Score, we have:

$$p_t(y|x_t) = \int_{x_0} p(y|x_0) p(x_0|x_t) dx_0$$

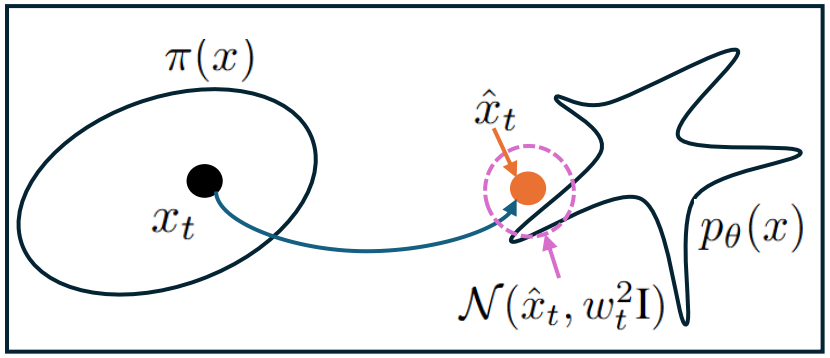
- We know $p(y|x_o) \sim \sigma_y \mathcal{N}(Ax_0, \mathrm{I})$.

How do we approximate $p(x_0|x_t)$?

Pseudoinverse-Guided Diffusion Models

- We use the Pseudoinverse-Guided Diffusion Model (ΠGDM) approximation[2].
- Use an isotropic Gaussian Approximation for $p(x_0|x_t) \sim \mathcal{N}(\hat{x}_t, w_t^2\mathbf{I})$

  where w_t^2 is a data dependent parameter.



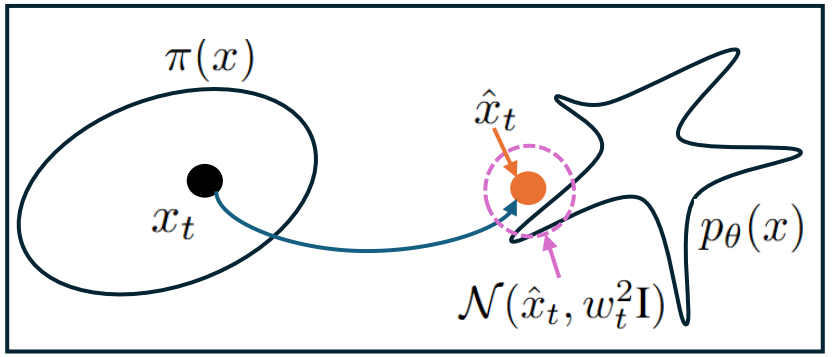
[2] Song et al. "Pseudoinverse-guided diffusion models for inverse problems." (ICLR), 2023

Pseudoinverse-Guided Diffusion Models

- We use the Pseudoinverse-Guided Diffusion Model (ΠGDM) approximation[2].
- Use an isotropic Gaussian Approximation for $p(x_0|x_t) \sim \mathcal{N}(\hat{x}_t, w_t^2\mathrm{I})$

  where w_t^2 is a data dependent parameter.
- Next, using the Gaussian Convolution theorem,

$$p_t(y|x_t) = \int_{x_0} p(y|x_0)p(x_0|x_t)dx_0$$
$$\sim \mathcal{N}(A\hat{x}_t, w_t^2 AA^\top + \sigma_y^2\mathrm{I})$$



- Finally, we can derive the likelihood score as:

$$\nabla_{x_t} \log p_t(y|x_t) \approx \underbrace{(y - A\hat{x}_t)(w_t^2 AA^\top + \sigma_y^2\mathrm{I})^{-1}A}_{\text{vector}} \underbrace{\frac{\partial \hat{x}_t}{\partial x_t}}_{\text{Jacobian}}$$

[2] Song et al. "Pseudoinverse-guided diffusion models for inverse problems." (ICLR), 2023

Roadmap

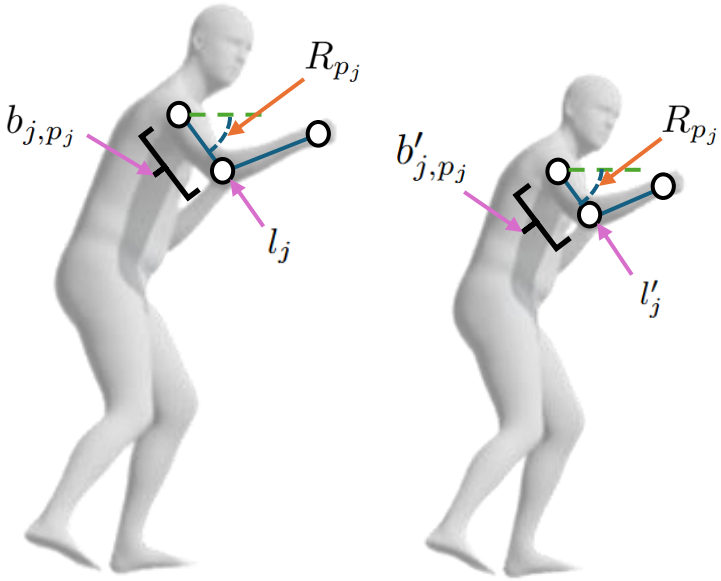
- Problem formulation
  - Kinematic tree
  - Scale dependent vs. scale free pose
  - 3-point pose estimation
- Learning representations
  - Rotation representation
  - 6D representation
- Background on Diffusion
  - Score function
  - Guided diffusion and Posterior sampling
  - Pseudo-inverse based posterior sampling (ΠGDM)
- InPose: Our work
  - Limitations of past work
  - Key intuition: learning scale free parameters
  - Diffusion priors and ΠGDM linear model
  - Theorem
  - InPose pipeline
  - Results

Limitations of Deep Learning Approaches

- E.g. Neural Network f_ϕ to predict pose:

$$\{l_M(i), r_M(i)\} = f_\phi(l_m(i), r_m(i))$$

- Issue – $\{l_j\}$ are a function of bones $\{b_{j,p_j}\}$
- Neural Network trained on one user's data can't generalize to a new user.



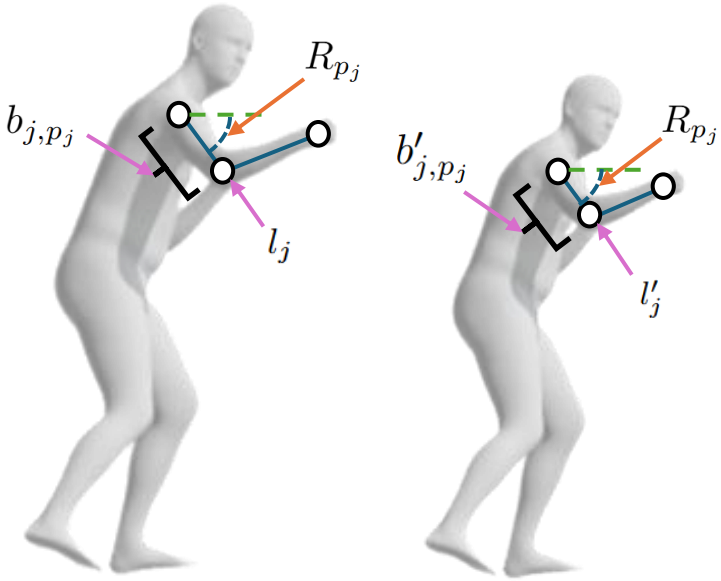
Can we reuse the same neural network without retraining?

Limitations of Deep Learning Approaches

- E.g. Neural Network f_ϕ to predict pose:

$$\{l_M(i), r_M(i)\} = f_\phi(l_m(i), r_m(i))$$

- Issue – $\{l_j\}$ are a function of bones $\{b_{j,p_j}\}$
- Neural Network trained on one user's data can't generalize to a new user.
- Use Guided Diffusion to overcome these issues.



Opportunity 1 – Use scale-free $r_m(i)$ for CFG

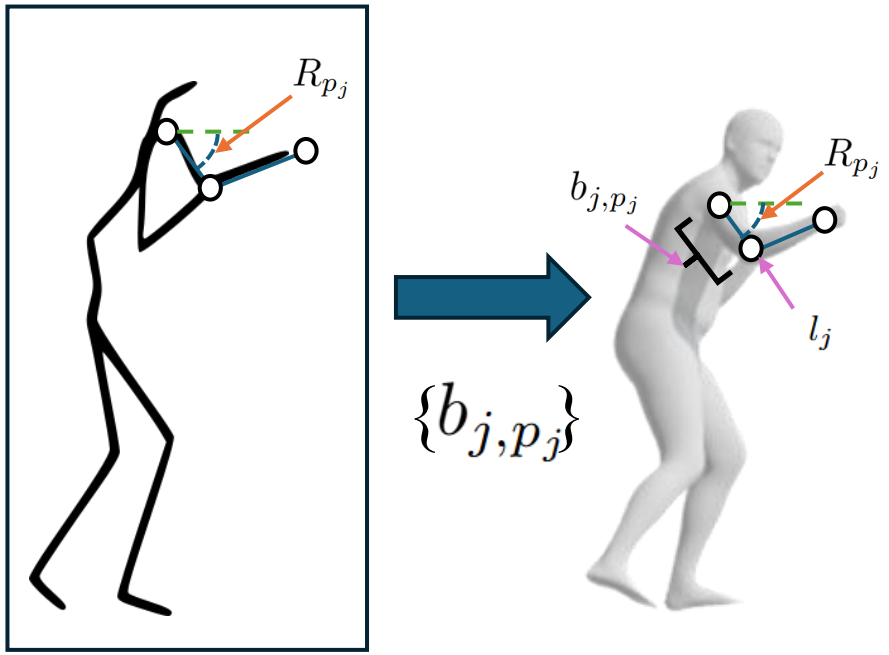
Opportunity 2 – Use ΠGDM to incorporate $l_m(i)$

Diffusion for 3-Point Pose Estimation

- Opportunity 3 – Predict the Scale-free Pose $\{R_j\}$

  .

- Use new user's $\{b_{j,p_j}\}$ to compute Scale-dependent Pose $\{l_j\}$.

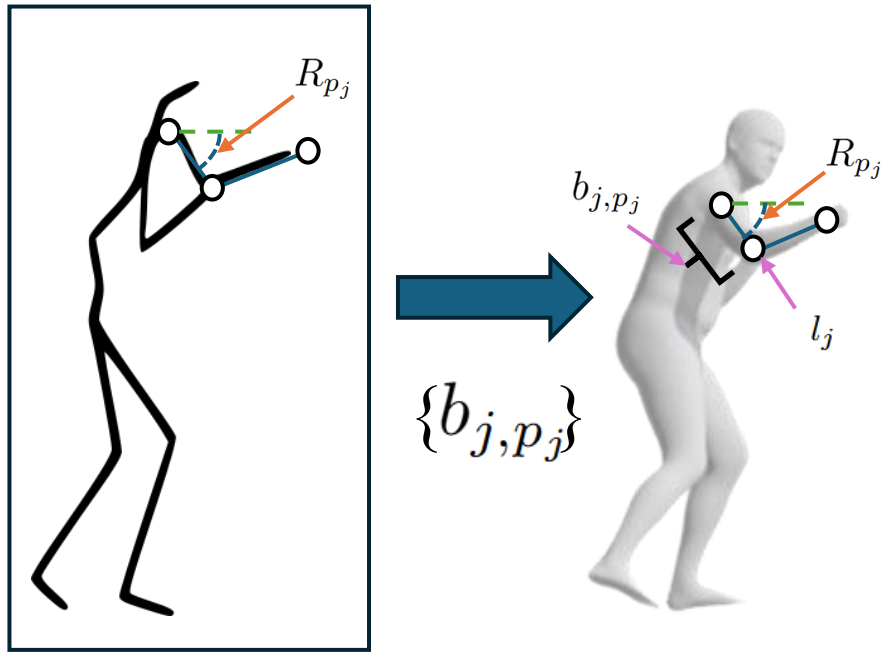


Diffusion for 3-Point Pose Estimation

- Opportunity 3 – Predict the Scale-free Pose $\{R_j\}$

  .

- Use new user's $\{b_{j,p_j}\}$ to compute Scale-dependent Pose $\{l_j\}$.



- We thus use the following score function:

$$\nabla_{r_M^t} \log p_t(r_M^t | l_m, r_m) = \underbrace{\nabla_{r_M^t} \log p_t(r_M^t | r_m)}_{\text{CFG Score}} + \underbrace{\nabla_{r_M^t} \log p_t(l_m | r_M^t, r_m)}_{\Pi\text{GDM Score}}$$

ΠGDM linear model

- Recall ΠGDM requires a linear measurement model $y = Ax_0 + n$ with $n \sim \sigma_y \mathcal{N}(0, \mathrm{I})$
- If the rotations $\{R_j\}$ were available to us, we can create the measurement model using Equation (1b) as follows:

$$[R_1 ... R_{p_j}] \cdot [b_{2,1}^\top ... b_{j,p_j}^\top]^\top = l_j$$

$$l_j = l_{p_j} + R_{p_j} \cdot b_{j,p_j} \qquad \text{(1b)}$$

ΠGDM linear model

- If the rotations $\{R_j\}$ were available to us, we can create the measurement model using Equation (1b) as follows:

$$[R_1 ... R_{p_j}] \cdot [b_{2,1}^\top ... b_{j,p_j}^\top]^\top = l_j$$

$$l_j = l_{p_j} + R_{p_j} \cdot b_{j,p_j} \qquad \text{(1b)}$$

- Setting $C = [R_1 ... R_{p_j}]$ and $\kappa = [b_{2,1}^\top ... b_{j,p_j}^\top]^\top$, we get:

$$(\mathrm{I}_3 \otimes \kappa^\top) \cdot \mathrm{vec}(C) = l_j$$

where $\otimes$ is the Kronecker product.

- Our linear measurement function is:

$$\mathcal{A} := \mathrm{I}_3 \otimes \kappa^\top$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} b_{2,1}^\top ... b_{j,p_j}^\top \end{bmatrix}}_{\mathcal{A}} \bullet \underbrace{\begin{bmatrix} \mathrm{vec}([R_1 & \cdots & R_{p_j}]) \end{bmatrix}}_{x_0} = \underbrace{\begin{bmatrix} l_j \end{bmatrix}}_{y}$$

ΠGDM linear model

- If the rotations $\{R_j\}$ were available to us, we can create the measurement model using Equation (1b) as follows:

$$[R_1 ... R_{p_j}] \cdot [b_{2,1}^\top ... b_{j,p_j}^\top]^\top = l_j$$

$$l_j = l_{p_j} + R_{p_j} \cdot b_{j,p_j} \qquad \text{(1b)}$$

- Setting $C = [R_1 ... R_{p_j}]$ and $\kappa = [b_{2,1}^\top ... b_{j,p_j}^\top]^\top$, we get:

$$(\mathrm{I}_3 \otimes \kappa^\top) \cdot \mathrm{vec}(C) = l_j$$

- Our linear measurement function is thus $\mathcal{A} := \mathrm{I}_3 \otimes \kappa^\top$
- But like most SoTA deep-learning models, we represent rotations as $r(R) \in \mathbb{R}^6$
- We need to recover the rotation matrix using the non-linear $\mathcal{D}(r)$

Can ΠGDM be used even after applying $\mathcal{D}()$?

Incorporating the Non-linearity

- Turns out, we can, if we apply the column normalization on the predicted r_M first.
- Let's term the CFG score model $\epsilon_\theta(r_M^t, t, r_m)$, where the *t* parameter is incorporated to inform the denoiser of the current noise level.
- We then have the following Theorem:

Theorem 1. *We are given a well-trained score model ϵ_θ, that learns the score function $\epsilon_\theta(r_M^t, t, r_m)$, and denoises $\hat{r}_M^t$. If the model ensures that $||\hat{r}_j^{t,1:3}|| = ||\hat{r}_j^{t,4:6}|| = 1$, $\langle \hat{r}_j^{t,1:3}, \hat{r}_j^{t,4:6} \rangle = 0$, $\forall j \in M$ then $p_t(\mathcal{D}(r_M^0)|r_M^t) \approx \mathcal{N}(\mathcal{D}(\hat{r}_M^t), w_t^2 \Sigma_{\hat{r}_M^t})$ where $\Sigma_{\hat{r}_M^t}$ is a positive definite matrix.*

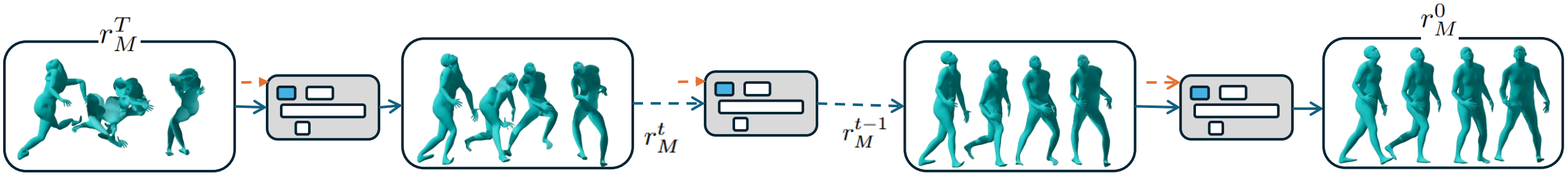
- In other words, we can use a modified Gaussian approximation over the isotropic Gaussian from the original ΠGDM algorithm.

InPose

- We can thus calculate the modified Likelihood score as follows:

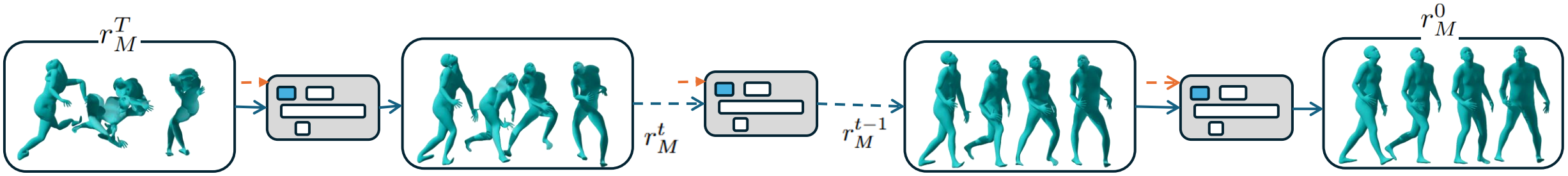
$$\nabla_{r_M^t} \log p_t(l_m | r_M^t, r_m) = \underbrace{((l_m - \mathcal{A} \cdot \mathcal{D}(\hat{r}_M^t))^\top (w_t^2 \mathcal{A} \Sigma_{\hat{r}_M^t} \mathcal{A}^\top + \sigma_l^2 \mathrm{I})^{-1} \mathcal{A}}_{\text{vector}} \underbrace{\frac{\partial \mathcal{D}(\hat{r}_M^t)}{\partial r_M^t})^\top}_{\text{Jacobian}}$$

InPose Pipeline

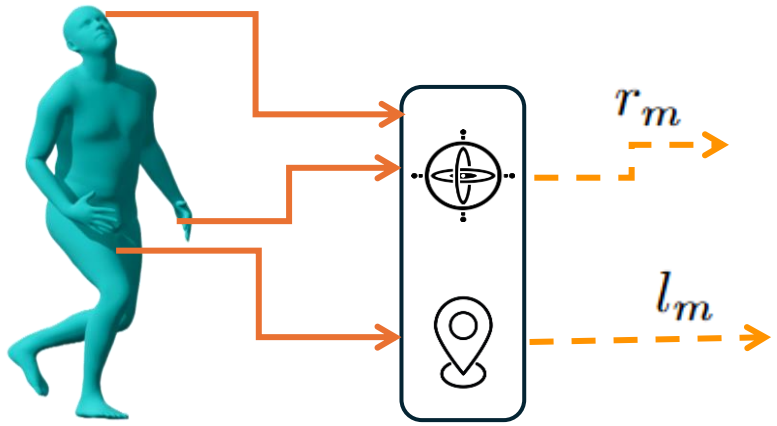


We use DDPM during training and DDIM during inference.

InPose Pipeline

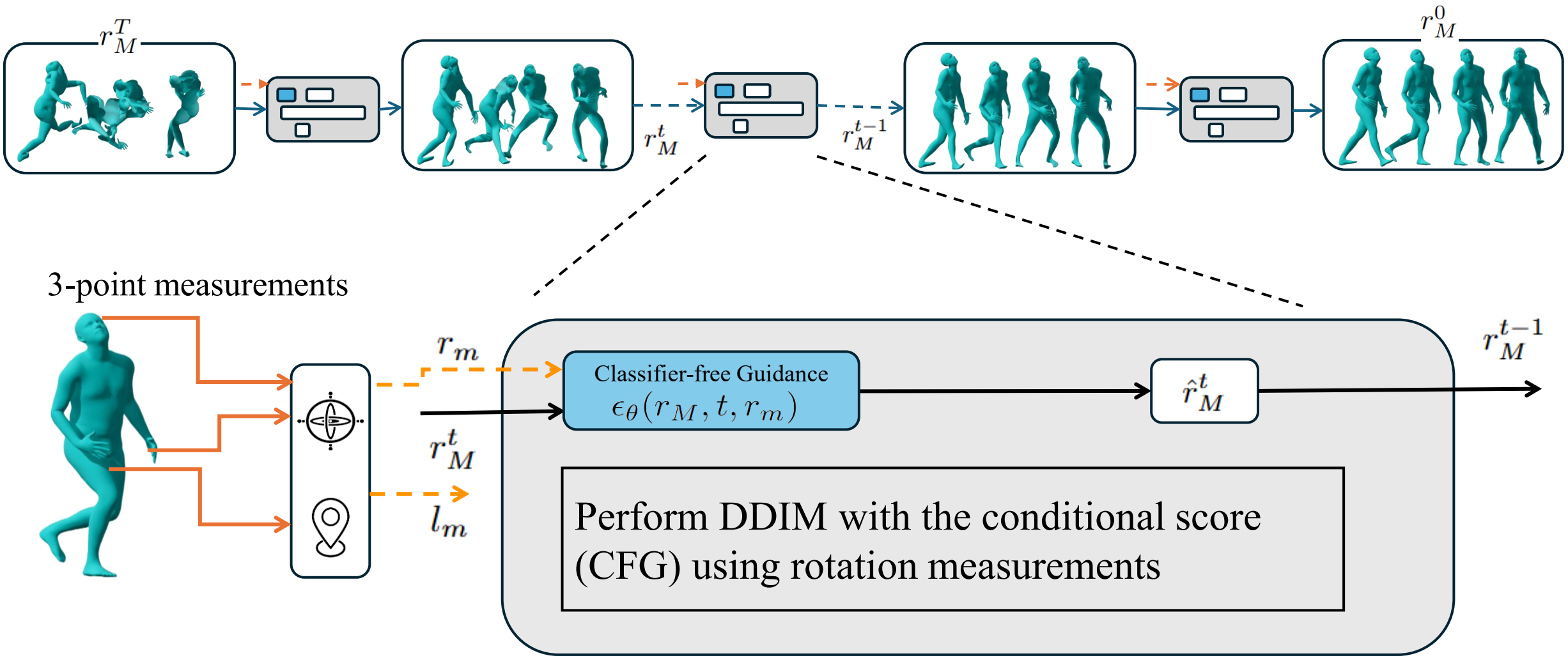


3-point measurements

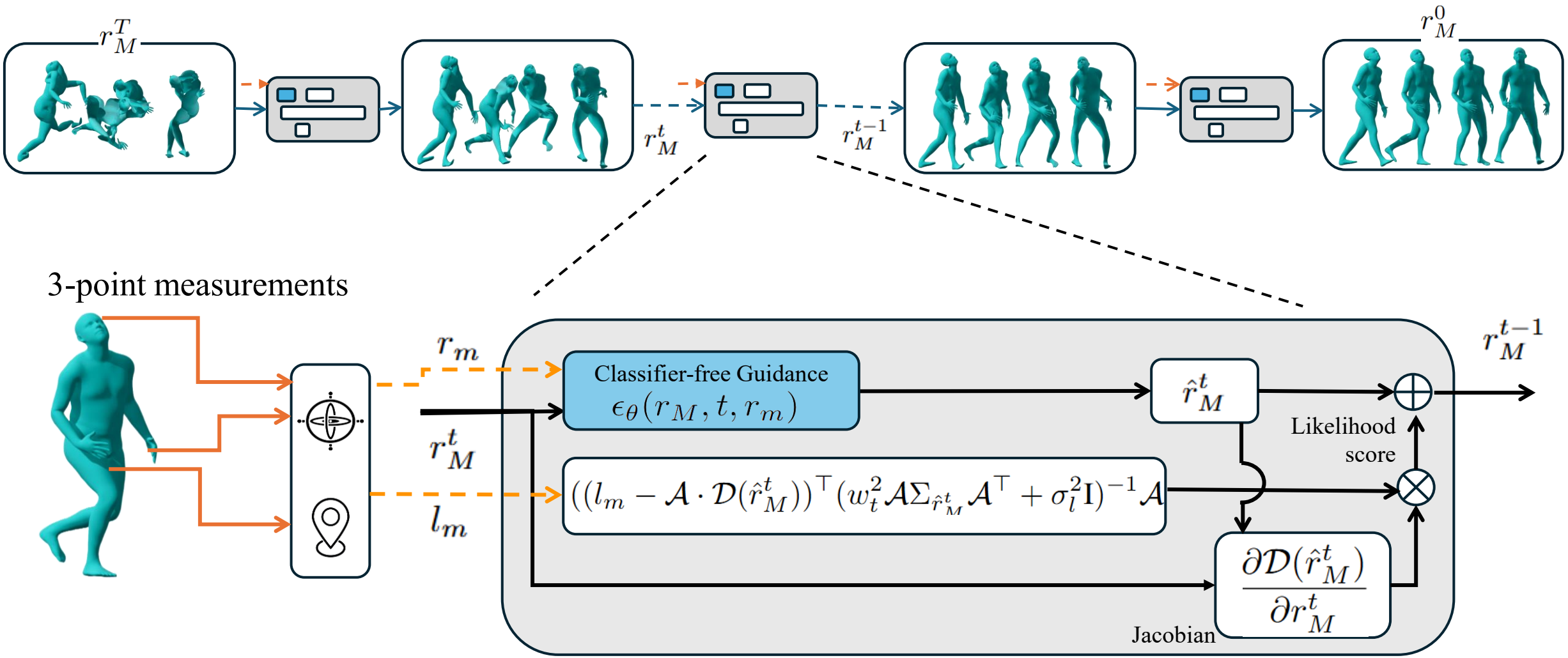


Capture location and rotation measurements from the head and wrists

InPose Pipeline

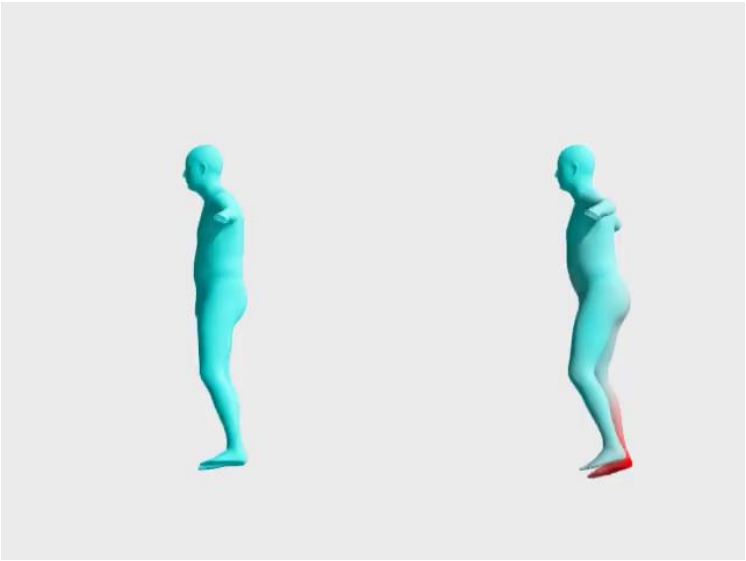


InPose Pipeline

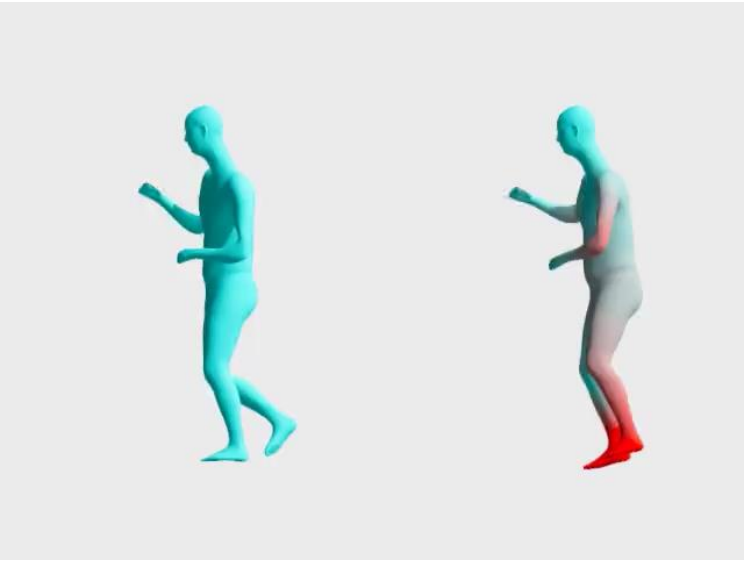


Results

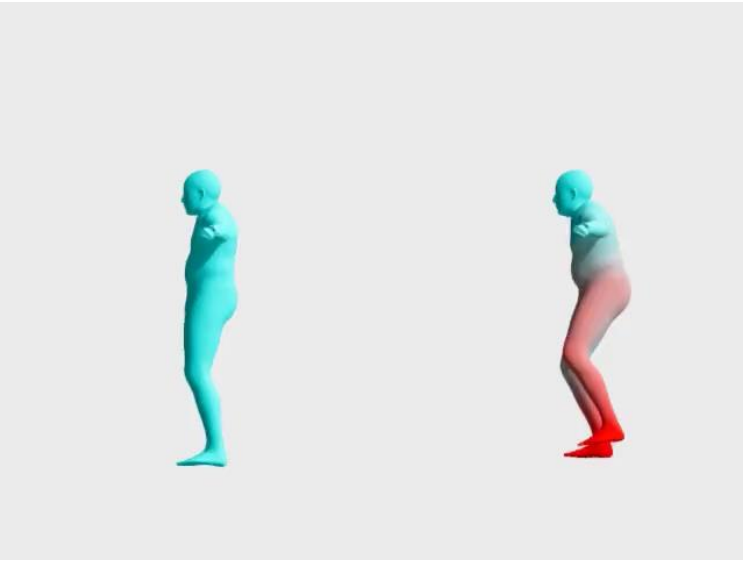
Default body shape



Tall body shape



Short body shape



InPose limitations

- Lower body movement – We don't have lower body measurements:
    - Option 1: infer the lower body movement from the current CFG score (prior-based).
    - Option 2: use the head joint location l_{head} for CFG.
- Differential Inputs – We can't directly use the joint positions in our linear system because we don't have access to the root location beforehand.
    - Use differential inputs such as $l_{\text{head}} - l_{\text{wrist}}$.
    - Likelihood is thus proportional to:

$$\nabla_{r_M^t} \log p_t(l_m | r_M^t, r_m) \propto ((l_{head} - l_{wrists}) - \mathcal{A}\hat{r}_M^t)$$