Lightweight Multi-View 3D Pose Estimation through Camera-Disentangled Representation

Edoardo Remelli

Shangchen Han

Sina Honari

Pascal Fua

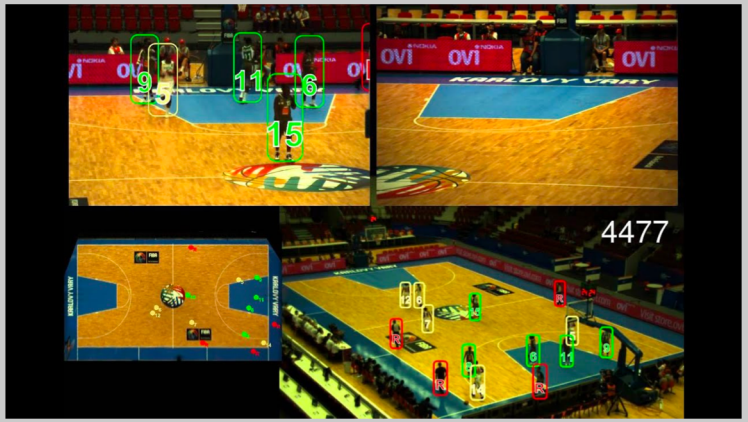
Robert Wang

EPFL

facebook Reality Labs

Motivation

Multi-view input from **synchronized and calibrated cameras**



State-of-the-art **multi-view pose estimation** solutions project 2D detections to 3D volumetric grids and reason jointly across views through **computationally intensive** 3D CNN or Pictorial Structures

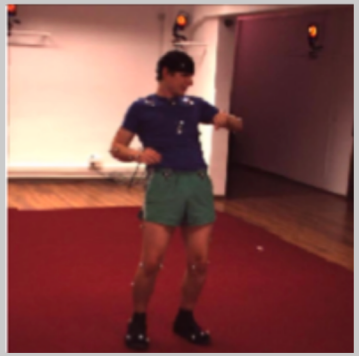
Can we fuse features both effectively and efficiently in latent space instead?

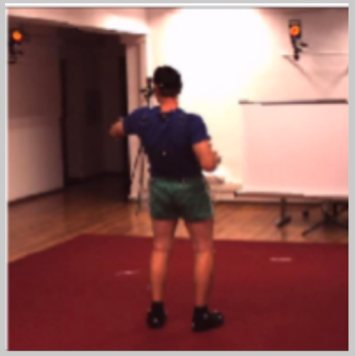
Problem Setting

Pinhole camera model: $x = P(X) = K\,E(X) = K(RX + t)$

Given:

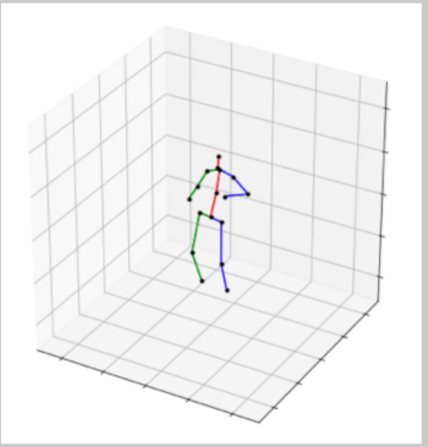
- Multi-view input crops $\{I_i\}_{i=1}^{N}$
- Camera projection matrices $\{P_i\}_{i=1}^{N}$





Find:

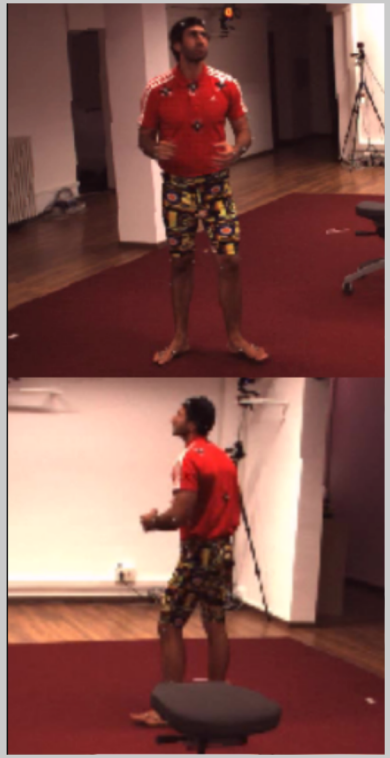
- 3D articulated pose x in world coordinates



Lightweight pose estimation

Real-time multi-view 3D pose estimation methods:

- Do not share information between features, although they represent the same pose in different coordinate systems
- Do not supervise for the metric of interest and use triangulation as a post-processing step



Resnet

Shallow decoder

Soft Integration

?

Resnet

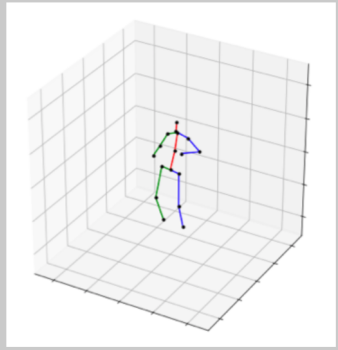
Shallow decoder

Soft Integration

2D detections

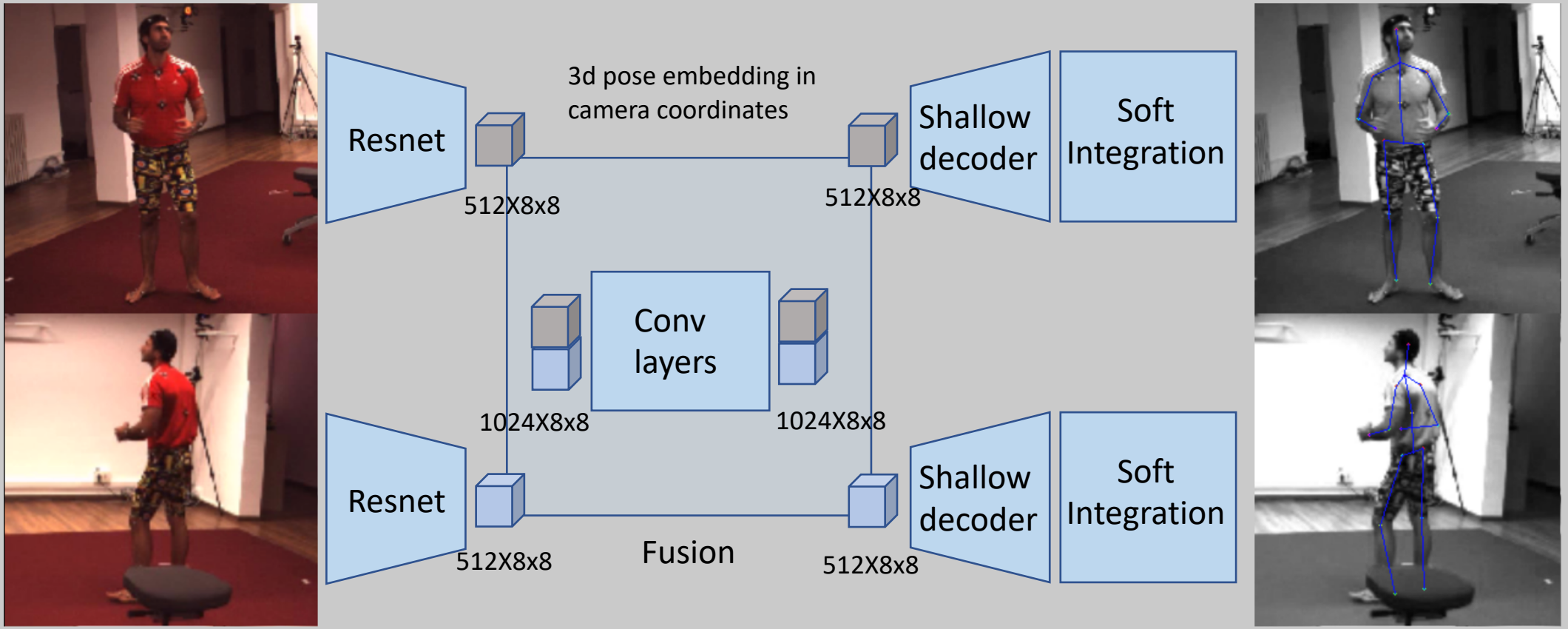
?

Triangulation



3D pose

Our Baseline [Fusion]



How to reason jointly about pose across views?

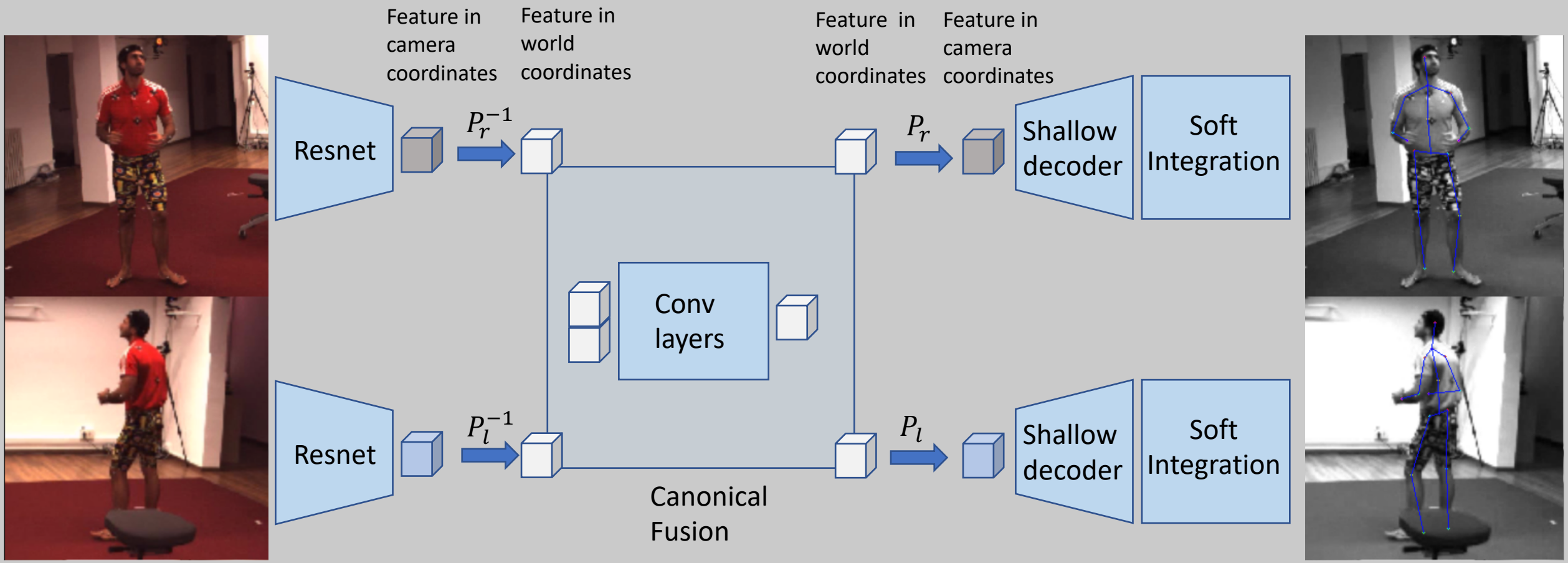
Let the network do all the hard work…

Pros: simple to implement, effective

Cons: overfits by design to camera setting, does not exploit camera transforms explicitly

Can we do better?

Pinhole camera model: $x = P(X) = K\,E(X) = K(RX + t)$



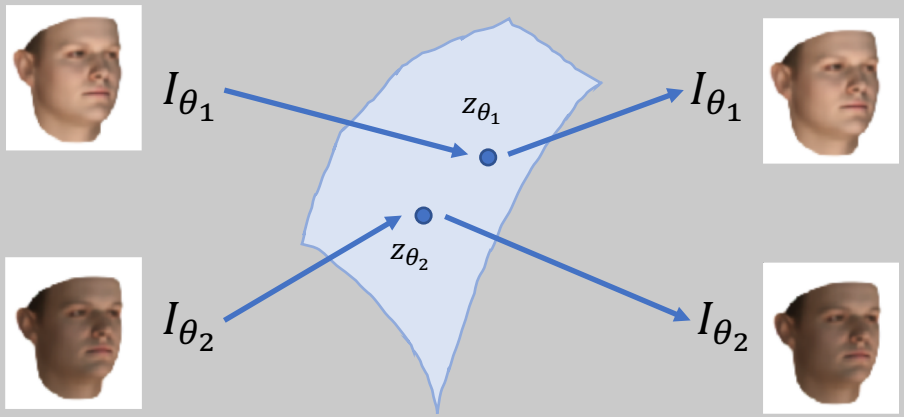
If we could map features to a common frame of reference before fusing them, jointly reasoning about views would become much easier for the network.

How to apply transformation to feature maps **without** using 3d volume aggregation?

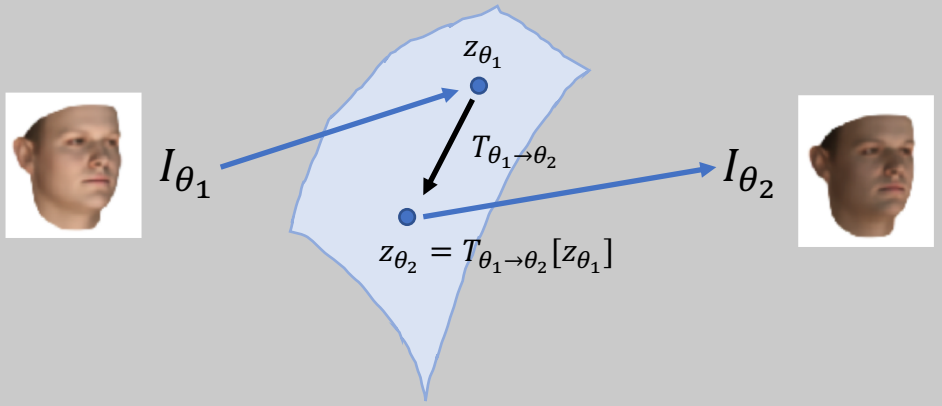
Review of Transforming Auto-Encoders

Given a representation learning task and a known source of variation θ, *[1]* proposes to learn equivariance with respect to the source of variation by conditioning latent code on the variation

Auto-Encoder



Transforming Auto-Encoder



How to choose transform $T_{\theta_1 \to \theta_2}$?

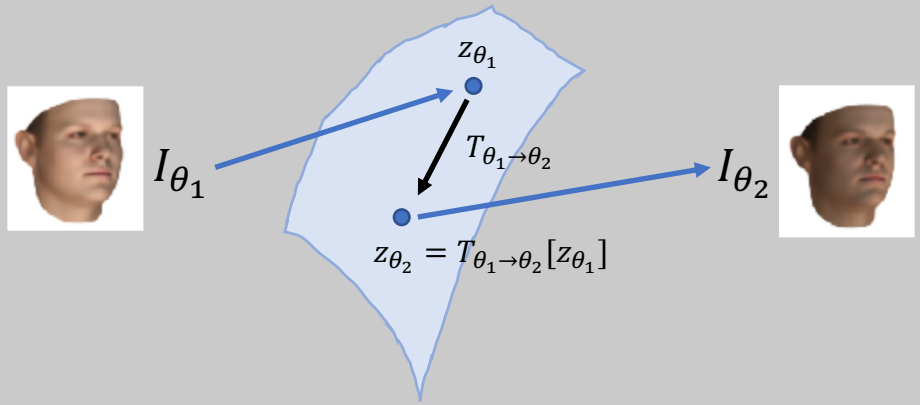
[1]: Worrall, Garbin, Turmukhambetov, and Brostow. Interpretable Transformations with Encoder-Decoder Networks

Review of Transforming Auto-Encoders [1]

How to choose transformation T?

- Linear
- Invertible
- Norm preserving

ROTATIONS



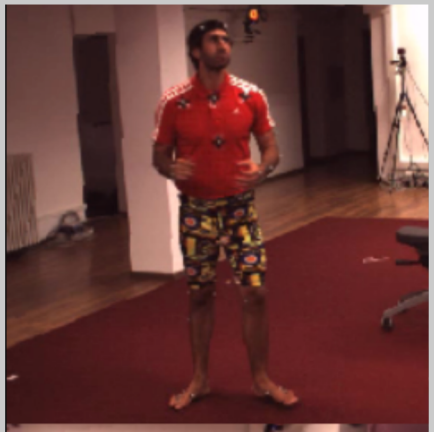
Feature transform layer:

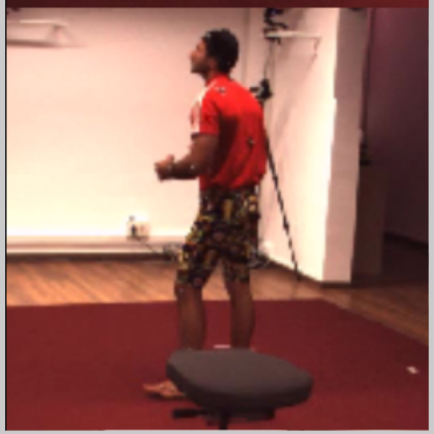
$$z_{\theta_1} \in \mathbb{R}^{F\ x\ W\ x\ H}$$
$$z_{\theta_1} = z_{\theta_1}.reshape(2, N)$$
$$z_{\theta_2} = R_{\theta_1 \to \theta_2} z_{\theta_1}$$
$$z_{\theta_2} = z_{\theta_2}.reshape(F, W, H)$$

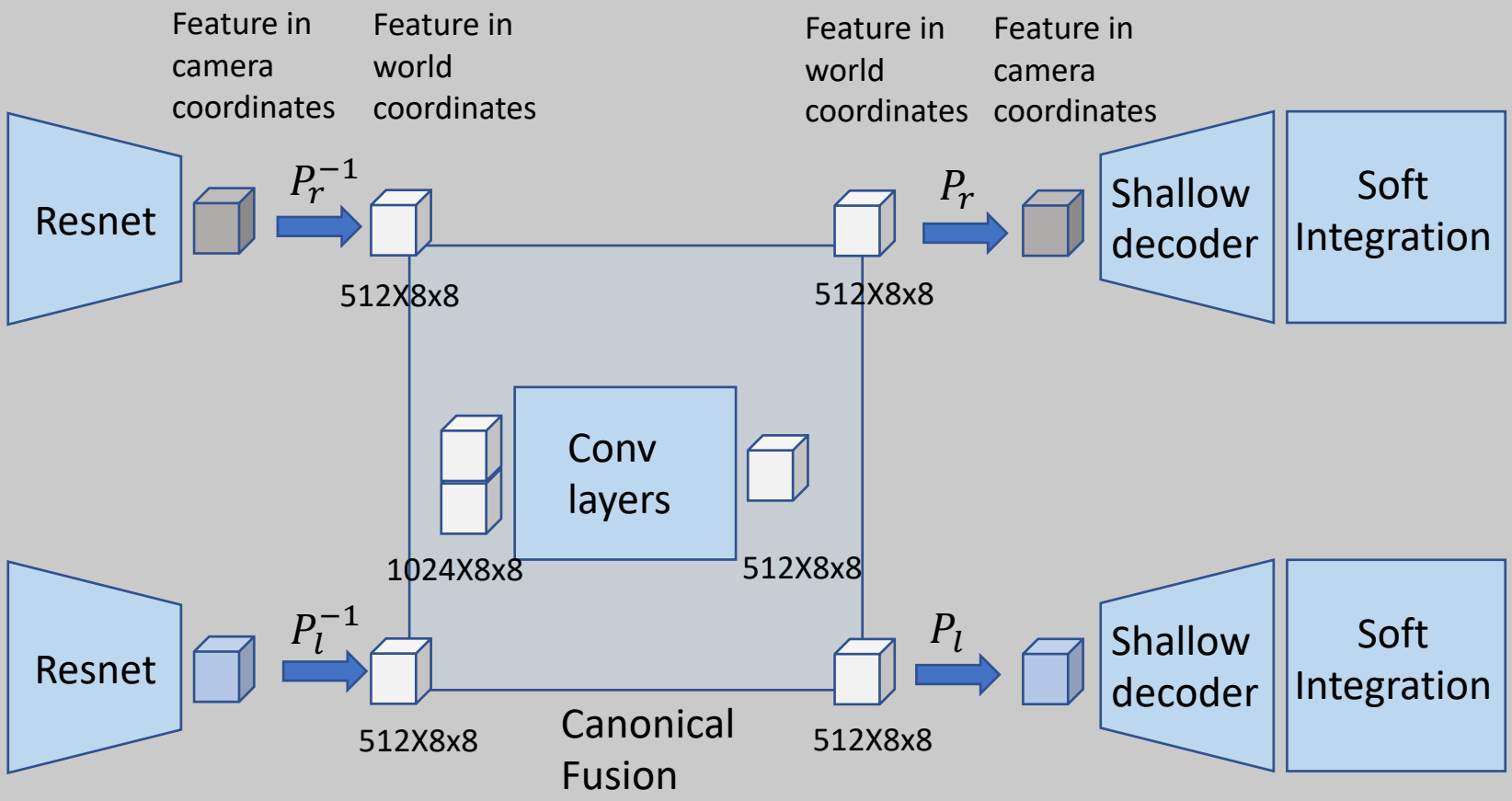
We can use a feature transform layers to map features between frames of reference

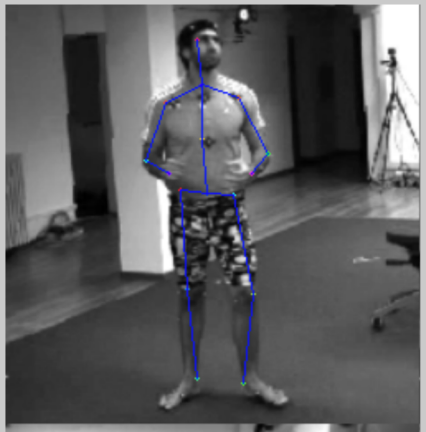
[1]: Worrall, Garbin, Turmukhambetov, and Brostow. Interpretable Transformations with Encoder-Decoder Networks

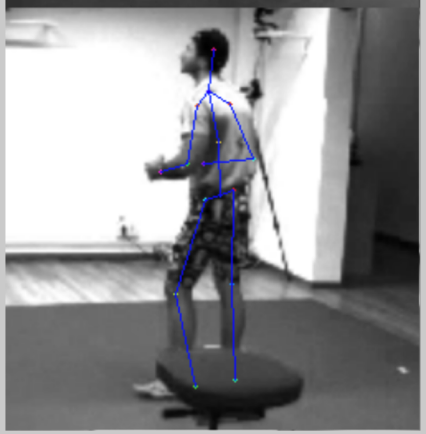
Our architecture [Canonical Fusion]







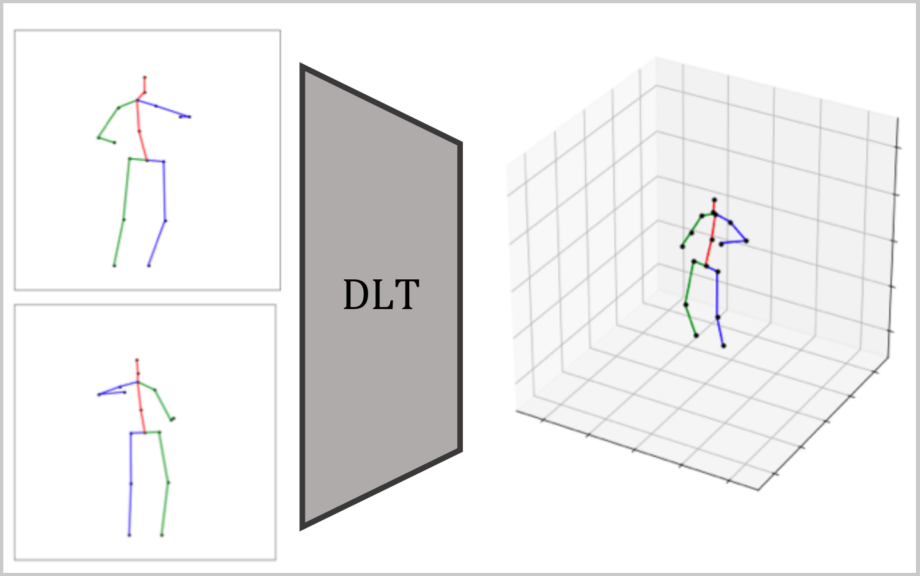




- Makes use of camera information (Flexible)
- Lightweight (Does not rely on volumetric aggregation)

Now that we computed 2D detections, how can we lift them to 3D differentiably?

Direct Linear Transform (DLT)

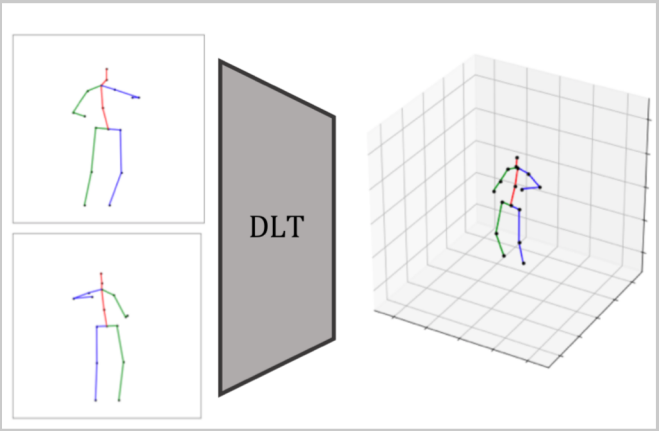


$\{\boldsymbol{u}_i\}_{i=1}^{N}$ $\boldsymbol{x}$

Review of DLT

From Epipolar Geometry:

$$d_i \boldsymbol{u}_i = P_i \boldsymbol{x} \rightarrow \begin{cases} d_i u_i = p_i^{1T} \boldsymbol{x} \\ d_i v_i = p_i^{2T} \boldsymbol{x} \\ d_i = p_i^{3T} \boldsymbol{x} \end{cases} \rightarrow \begin{cases} (p_i^{3T} u_i - p_i^{1T}) \boldsymbol{x} = \boldsymbol{0} \\ (p_i^{3T} v_i - p_i^{2T}) \boldsymbol{x} = \boldsymbol{0} \end{cases}$$



$\{\boldsymbol{u}_i\}_{i=1}^N$ $\boldsymbol{x}$

Accumulating over available N views:

$$A\boldsymbol{x} = \boldsymbol{0}, \qquad A \in \mathbb{R}^{2N \times 4}$$

Admits non-trivial solution only if $\boldsymbol{u}_i$ and P_i are not noisy, therefore we must solve a relaxed version

$$\min_{\mathrm{x}} |A\boldsymbol{x}|, \quad s.t. \ |x| = 1 \rightarrow$$

Equivalent to finding the eigenvector of $A^T A$ associated to the smallest eigenvalue

$$\lambda_{min}(A^T A)$$

How to solve it?

$$\lambda_{min}(A^T A)?$$

In literature, the smallest eigenvalue is found by computing a Singular Value Decomposition (SVD) of matrix A *[2]*

We argue that this is sub-optimal because:

- we need only the smallest eigenvalue, not full SVD factorization
- SVD is not a GPU friendly algorithm *[3]*

[2]: Hartley and Zisserman, Multiple view geometry in computer vision
[3]: Dongarra, Gates, Haidar, Kurzak, Luszczek, Tomov, and Yamazaki, Accelerating numerical dense linear algebra calculations with GPUs

How to solve it?

Step 1: derive a bound for the the smallest singular value of matrix A:

Theorem 1 *Let A be the DLT matrix associated to the non-perturbed case, i.e. $\sigma_{min}(A) = 0$. Let us assume i.i.d Gaussian noise $\varepsilon = (\varepsilon_u, \varepsilon_v) \sim \mathcal{N}(0, s^2 I)$ in our 2d observations, i.e. $(u^*, v^*) = (u + \varepsilon_u, v + \varepsilon_v)$, and let us denote as A^* the DLT matrix associated to the perturbed system. Then, it follows that:*

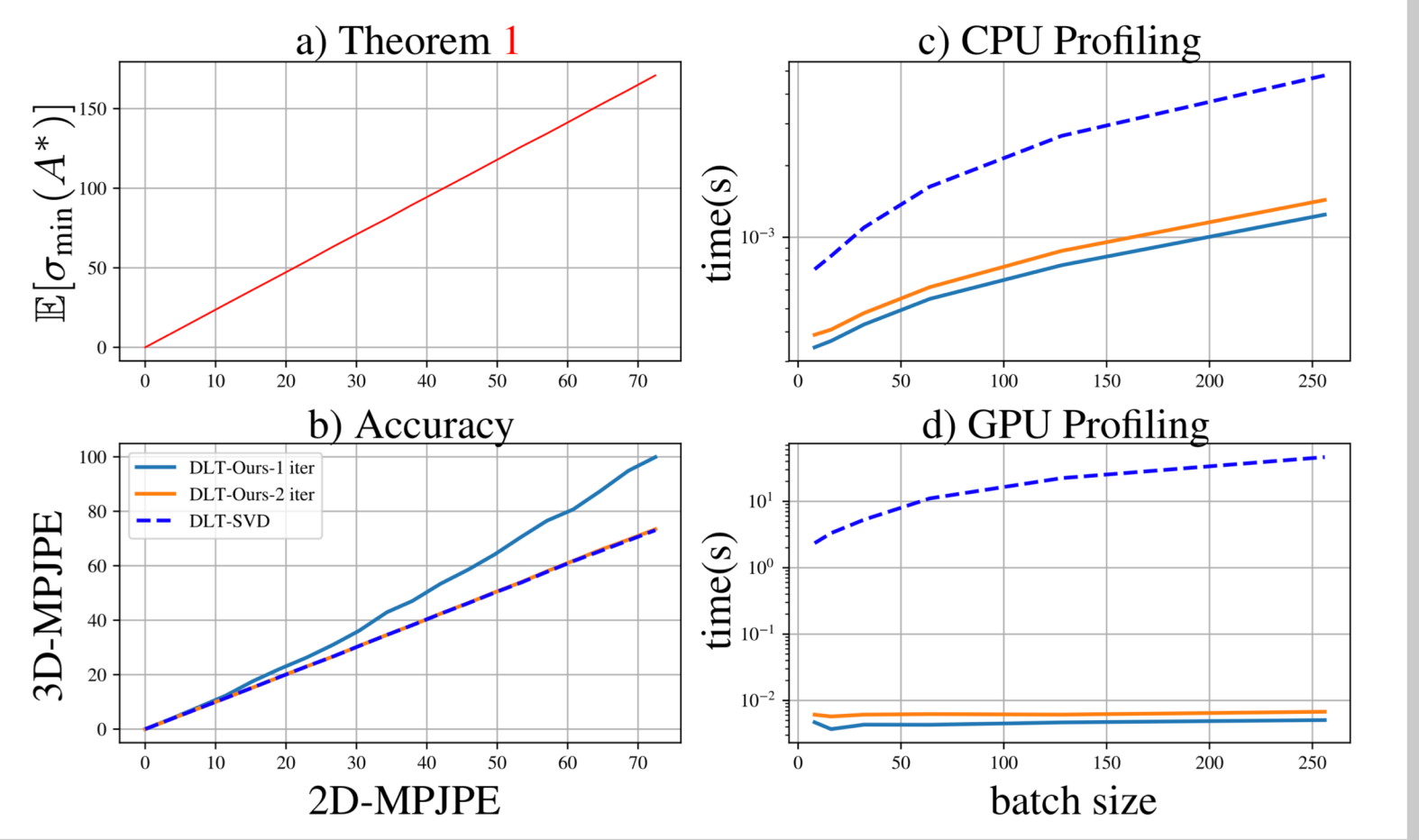
$$0 \leq \mathbb{E}[\sigma_{min}(A^*)] \leq Cs, \text{ where } C = C(\{u_i, P_i\}_{i=1}^N)$$

Step 2: use it to estimate the smallest singular value. Then refine the estimate iteratively using Shifted Power Iteration method. Algorithm 1 is guaranteed to converge to the desired singular value because of the bound above.

Algorithm 1: DLT-SII$(\{\mathbf{u}_i, P_i\}_{i=1}^N, T = 2)$

$A \leftarrow A(\{\mathbf{u}_i, P_i\}_{i=1}^N)$;
$B \leftarrow (A^T A + \sigma I)^{-1}$;
$\sigma \leftarrow 0.001$ (see Theorem 1);
$\mathbf{x} \leftarrow \text{rand}(4, 1)$;
for $i = 1 : T$ **do**
  $\mathbf{x} \leftarrow B\mathbf{x}$;
  $\mathbf{x} \leftarrow \mathbf{x}/\|\mathbf{x}\|$;
end
return $\mathbf{y} \leftarrow \mathbf{x}(0:3)/\mathbf{x}(4)$;

Quantitative Evaluation – Direct Linear Triangulation



For reasonably accurate 2D detections, our algorithm converges in as little as 2 iterations to the desired eigenvalue. Since it requires only a small matrix inversion and few matrix multiplications, it is much faster than performing full SVD factorizations, especially on GPUs.

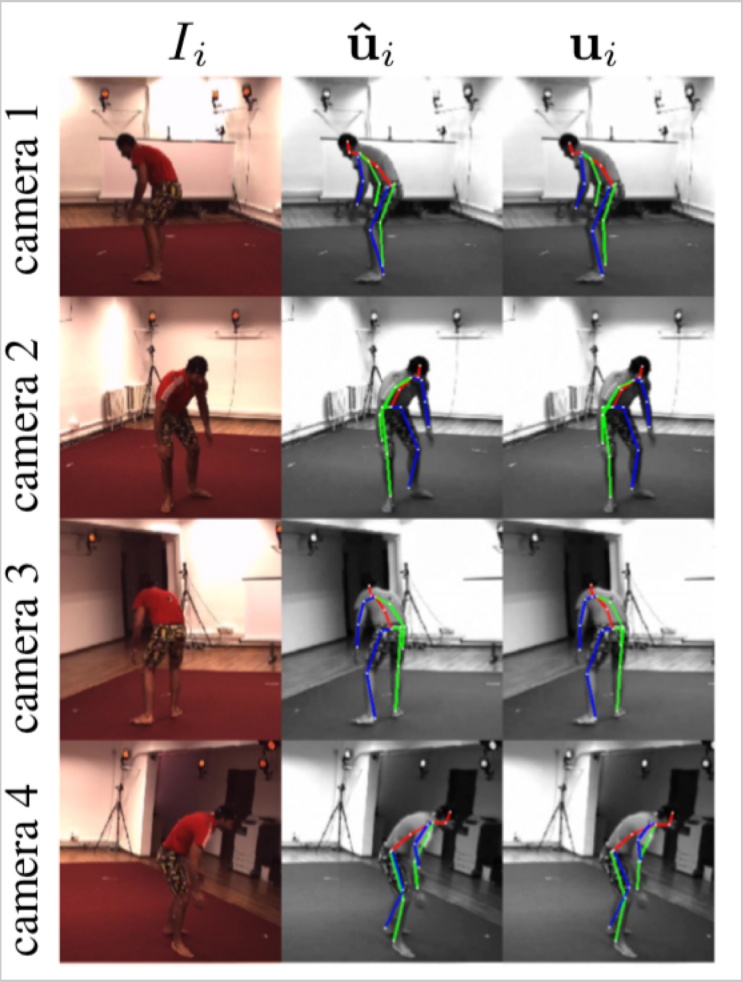
Quantitative Evaluation – H36M

w/o additional training data:

Methods	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Mean
Martinez *et al.* [20]	46.5	48.6	54.0	51.5	67.5	70.7	48.5	49.1	69.8	79.4	57.8	53.1	56.7	42.2	45.4	57.0
Pavlakos *et al.* [22]	41.2	49.2	42.8	43.4	55.6	46.9	40.3	63.7	97.6	119.0	52.1	42.7	51.9	41.8	39.4	56.9
Tome *et al.* [28]	43.3	49.6	42.0	48.8	51.1	64.3	40.3	43.3	66.0	95.2	50.2	52.2	51.1	43.9	45.3	52.8
Kadkhodamohammadi *et al.* [16]	39.4	46.9	41.0	42.7	53.6	54.8	41.4	50.0	59.9	78.8	49.8	46.2	51.1	40.5	41.0	49.1
Qiu *et al.* [23]	34.8	35.8	32.7	33.5	34.5	38.2	29.7	60.7	53.1	35.2	41.0	41.6	31.9	31.4	34.6	38.3
Qui *et al.* [23] + RPSM	28.9	32.5	26.6	28.1	**28.3**	**29.3**	**28.0**	36.8	41.0	**30.5**	35.6	30.0	28.3	**30.0**	30.5	31.2
Ours, Baseline	39.1	46.5	31.6	40.9	39.3	45.5	47.3	44.6	45.6	37.1	42.4	46.7	34.5	45.2	64.8	43.2
Ours, Fusion	31.3	37.3	29.4	29.5	34.6	46.5	30.2	43.5	44.2	32.4	35.7	33.4	31.0	38.3	32.4	35.4
Ours, Canonical Fusion (no DLT)	31.0	35.1	28.6	29.2	32.2	34.8	33.4	32.1	35.8	34.8	33.3	32.2	29.9	35.1	34.8	32.5
Ours, Canonical Fusion	**27.3**	**32.1**	**25.0**	**26.5**	29.3	35.4	28.8	**31.6**	**36.4**	31.7	**31.2**	**29.9**	**26.9**	33.7	**30.4**	**30.2**

w additional training data:

Methods	Model size	Inference Time	MPJPE
Qui *et al.* [23] Fusion + RPSM	2.1GB	8.4s	26.2
Iskakov *et al.* [15] Algebraic	320MB	2.00s	22.6
Iskakov *et al.* [15] Volumetric	643MB	2.30s	**20.8**
Ours, Baseline	**244MB**	**0.04s**	34.2
Ours, Canonical Fusion	251MB	**0.04s**	21.0



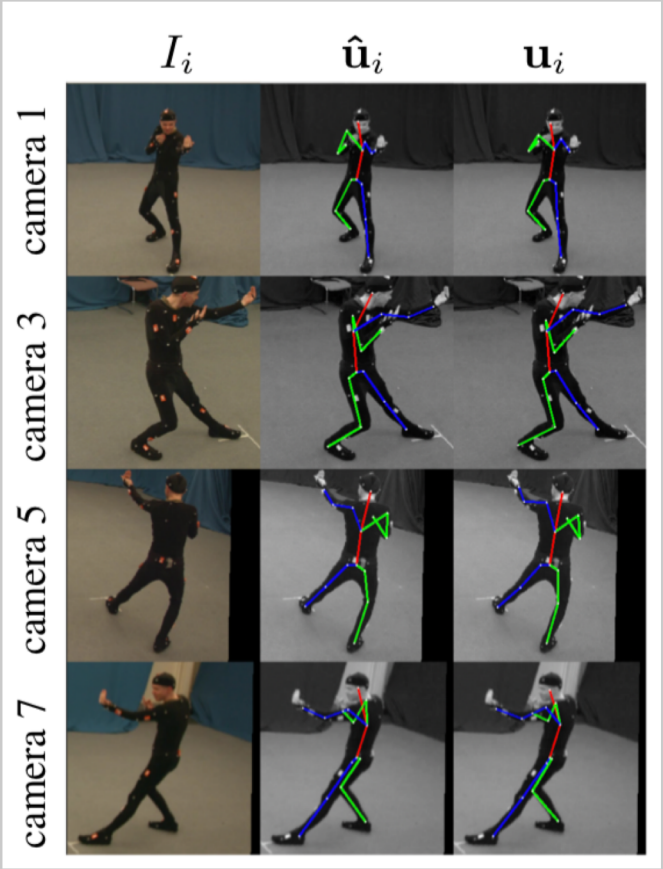
Quantitative Evaluation- Total Capture

Seen cameras:

Methods	Seen Subjects (S1,S2,S3)			Unseen Subjects (S4,S5)			Mean
	Walking	Freestyle	Acting	Walking	Freestyle	Acting	
Qui *et al.* [22] Baseline + RPSM	28	42	30	45	74	46	41
Qui *et al.* [22] Fusion + RPSM	19	**28**	21	32	**54**	**33**	29
Ours, Baseline	31.8	36.4	24.0	43.0	75.7	43.0	39.3
Ours, Fusion	14.6	35.3	20.7	28.8	71.8	37.3	31.8
Ours, Canonical Fusion(no DLT)	10.9	32.2	16.7	27.6	67.9	35.1	28.6
Ours, Canonical Fusion	**10.6**	30.4	**16.3**	**27.0**	65.0	34.2	**27.5**

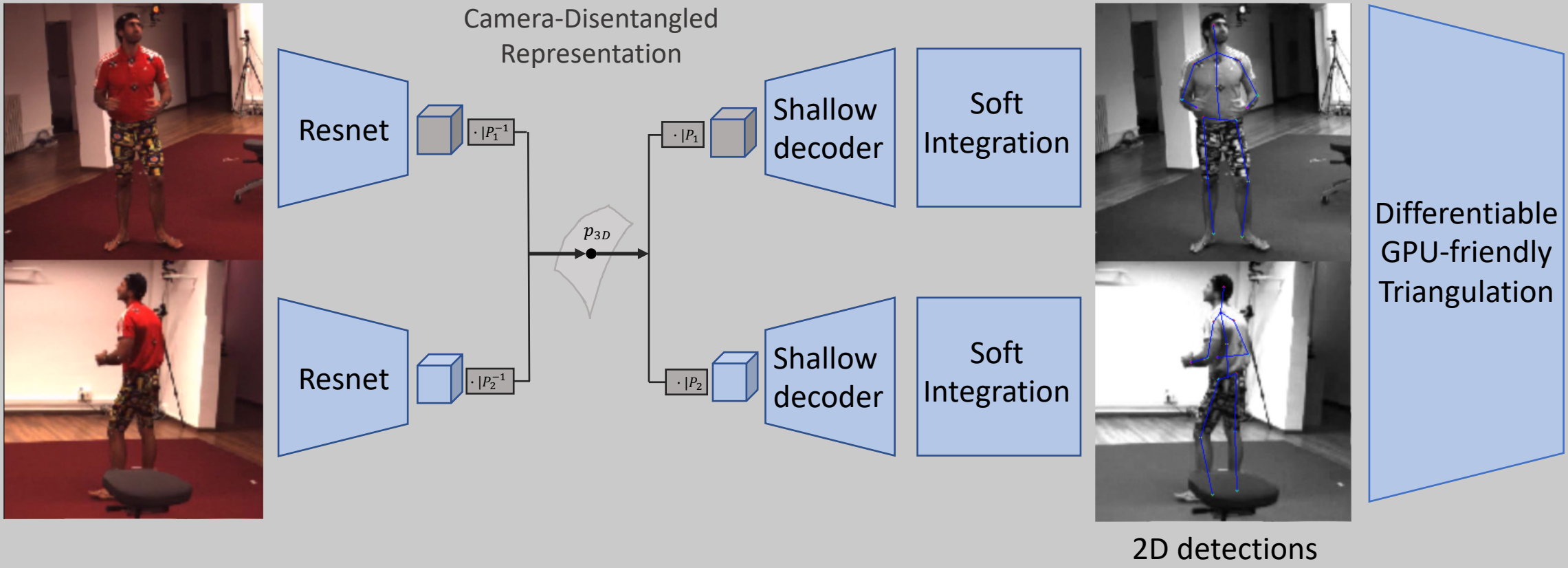
Unseen cameras:

Methods	Seen Subjects (S1,S2,S3)			Unseen Subjects (S4,S5)			Mean
	Walking	Freestyle	Acting	Walking	Freestyle	Acting	
Ours, Baseline	28.9	53.7	42.4	46.7	75.9	51.3	48.2
Ours, Fusion	73.9	71.5	71.5	72.0	108.4	58.4	78.9
Ours, Canonical Fusion	**22.4**	**47.1**	**27.8**	**39.1**	**75.7**	**43.1**	**38.2**



Contributions

- A novel multi-camera fusion technique that exploits 3D geometry in latent space to jointly reason about different views efficiently
- A new GPU-friendly differentiable algorithm for solving Direct Linear Triangulation, which is up to 3 orders of magnitude faster than SVD-based implementations while allowing us to supervise directly for the metric of interest



Please refer to the video for qualitative results and visualizations!

For any question, feel free to reach out to

edoardo.remelli@epfl.ch

Thank you!