3DV 2024

DANIEL KIENZLE, JULIAN LORENZ, KATJA LUDWIG, RAINER LIENHART

Universität Augsburg University

TOWARDS LEARNING MONOCULAR 3D OBJECT LOCALIZATION FROM 2D LABELS USING THE PHYSICAL LAWS OF MOTION

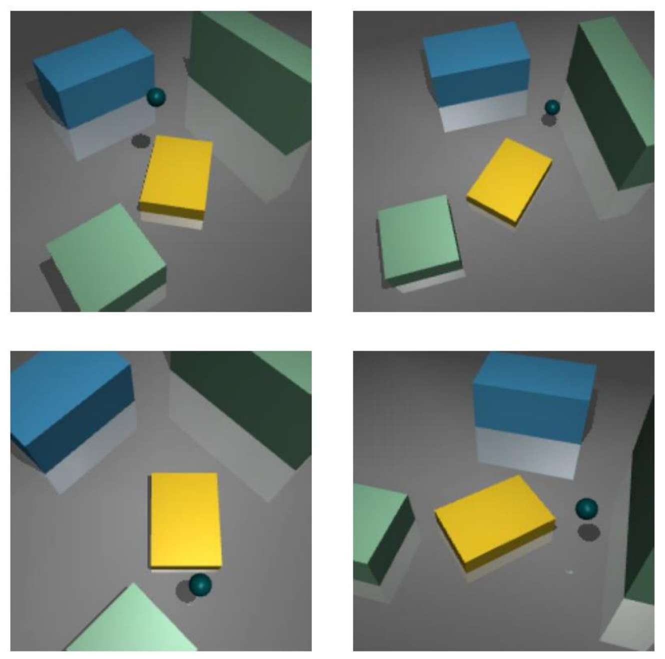
MOTIVATION

- Sports broadcasting companies are greatly interested in obtaining 3D information about the ball's location [1].
- Commercial technologies (e.g. Hawk-Eye [2] and View 4D [3]) rely on triangulation techniques → Expensive hardware is needed.
- Alternative approach: **Predict the ball's 3D position with neural networks** → Expensive 3D ground truth is usually needed for training.

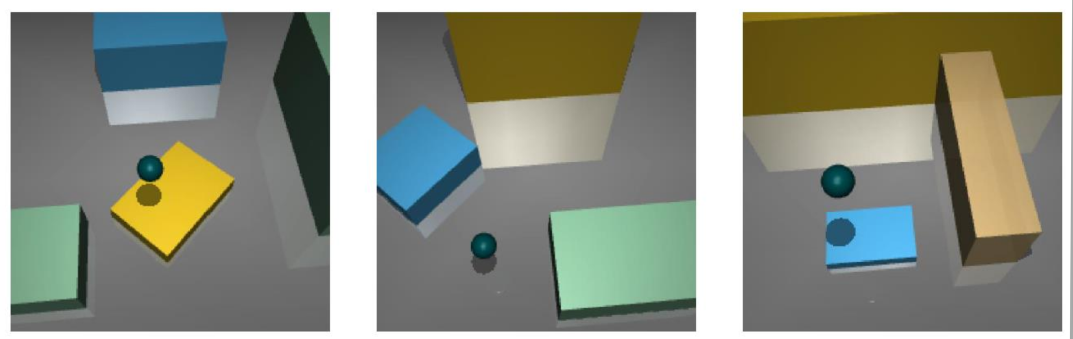
→ We train a neural network **without relying on 3D ground truth** annotations.

TASK

- We train a neural network for **precise 3D object localization in single images** from a single calibrated camera.
- **No expensive 3D labels** needed → utilize the **physical laws of motion** + easy-to-obtain **2D labels.**
- Infer **the latent third dimension**, even though this information is never seen during training.
- Physical motion can even be described in intricate situations (e.g. bouncing).
- Evaluation on both **synthetic and real-world datasets**. The datasets are provided for public use.



(a) *SD*: images captured from different camera locations in the 1st environment.



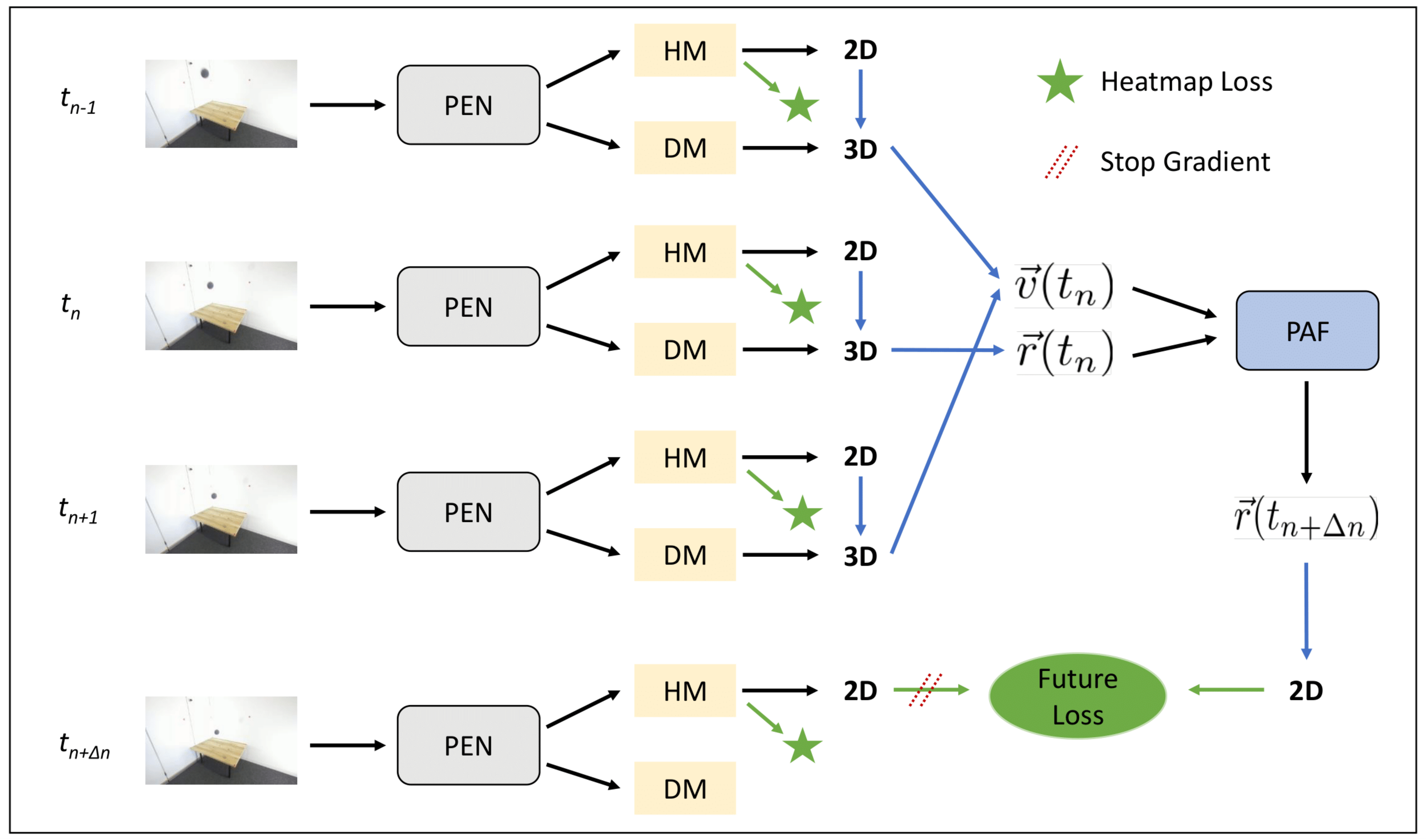
(b) *SD*: images captured in different environments.

(c) *RD*: images captured from different camera locations.

References:

- [1]: Gabriel Van Zandycke et. al., *Deepsportradar-v1: Computer vision dataset for sports understanding with high quality annotations*, 5th International ACM Workshop on Multimedia Content Analysis in Sports, 2022
- [2]: Hawk-Eye Innovations, *https://www.hawkeyeinnovations.com* (Accessed: February 26th 2024)
- [3]: Vieww GmbH, *https://vieww.com* (Accessed: February 26th 2024)
- [4]: J.R. Dormand and P.J. Prince, *A family of embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics, 1980

METHOD



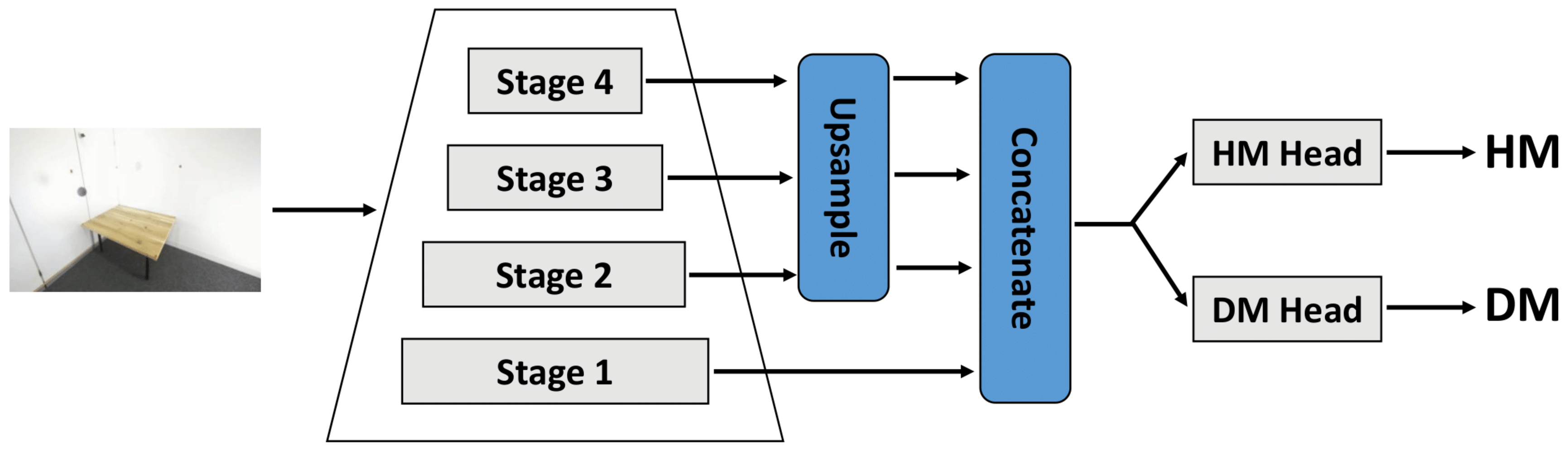
- **Two main modules**:
  1. **Position Estimation Network (PEN)**:
     - Neural network with learnable parameters.
     - Input: Single images; Output: Heatmap (*HM*) + depthmap (*DM*).

     → Obtain **world-coordinates** (using the camera matrices).
  2. **Physics Aware Forecasting Module (PAF):**
     - Solves the differential equations of motion.
     - No learnable parameters.

     → Obtain **world-coordinates and velocity at time t_n**.
- Apply PEN to images at time t_{n-1}, t_n and t_{n+1}:
  → Obtain **world-coordinates and velocity at time t_n**.
- **Loss:**

$$L = \|\vec{r}^{\,(I)}_{\mathrm{PEN}}(t_{n+\Delta n}) - \vec{r}^{\,(I)}_{\mathrm{PAF}}(t_{n+\Delta n})\|_{L1} + \frac{1}{|\mathcal{T}|}\sum_{t_i\in\mathcal{T}}\|HM(t_i) - HM_{\mathrm{gt}}(t_i)\|_{L2}$$

POSITION ESTIMATION NETWORK (PEN)



- Obtain **image-coordinates** ($x^{(I)}$ and $y^{(I)}$) from heatmap and **camera-depth** $z^{(C)}$ from depthmap.
- Use intrinsic and extrinsic camera matrices to obtain world-coordinates:

$$\begin{pmatrix}x^{(C)}\\y^{(C)}\\z^{(C)}\end{pmatrix} = \mathbf{M}_{\mathrm{int}}^{-1}\cdot\begin{pmatrix}x^{(I)}z^{(C)}\\y^{(I)}z^{(C)}\\z^{(C)}\end{pmatrix}, \quad \begin{pmatrix}x^{(W)}\\y^{(W)}\\z^{(W)}\\1\end{pmatrix} = \mathbf{M}_{\mathrm{ext}}^{-1}\cdot\begin{pmatrix}x^{(C)}\\y^{(C)}\\z^{(C)}\\1\end{pmatrix}$$

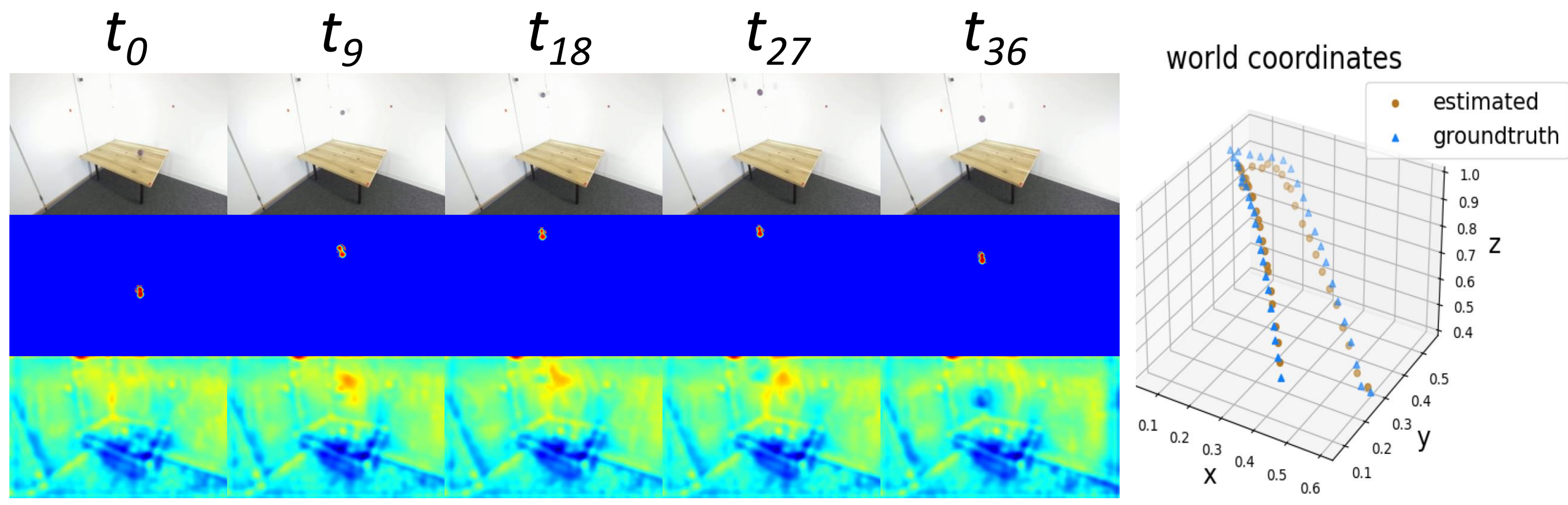
PHYSICS AWARE FORECASTING MODULE (PAF)

- The environment and relevant physics is described by the Hamiltonian *H*.
- Use a standard differential equation solver (e.g. [4]) to solve the differential equations of motion:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{r}^{\,(W)} = \frac{1}{m}\frac{\mathrm{d}}{\mathrm{d}\vec{v}}\mathcal{H}\,, \qquad m\frac{\mathrm{d}}{\mathrm{d}t}\vec{v} = -\frac{\mathrm{d}}{\mathrm{d}\vec{r}^{\,(W)}}\mathcal{H}$$

→ Predict the **world-coordinates at time $t_{n+\Delta n}$**.

EXPERIMENTS



- Metric: $DtG = \frac{1}{|\mathcal{V}|}\sum_{t_i\in\mathcal{V}}\|\vec{r}^{\,(C)}_{\mathrm{gt}}(t_i) - \vec{r}^{\,(C)}_{\mathrm{PEN}}(t_i)\|_{L2}$
- Generalization to unseen camera positions:

training set	*DtG* ± Δ*DtG* (cm) ↓ camera 1	camera 7	camera 8	camera 9
SD-S	22 ± 19	-	-	-
SD-M	19 ± 10	27 ± 23	23 ± 9	21 ± 10
SD-L	11 ± 6	28 ± 25	15 ± 8	16 ± 7

- Method works also for real-world data:

training set	*DtG* ± Δ*DtG* (cm) ↓ camera 1	camera 2
RD	7 ± 4	6 ± 4