Human Detection and Segmentation via Multi-view Consensus Supplementary Material

Isinsu Katircioglu[1] Helge Rhodin[2] Jörg Spörri[3] Mathieu Salzmann[1,4] Pascal Fua[1]
[1]EPFL, Lausanne, Switzerland [2]UBC, Vancouver, Canada
[3]Balgrist University Hospital, Zurich, Switzerland [4]ClearSpace SA, Lausanne, Switzerland
{firstname.lastname}@epfl.ch, rhodin@cs.ubc.ca, joerg.spoerri@balgrist.ch

In this document, we present the details of our differentiable multi-view consistency formulation, architecture design, and training strategies. Furthermore, we show additional qualitative results on the **Ski-PTZ**, **H36M** and **Handheld190k** datasets. We provide a video which can be accessed on https://youtu.be/bg3AYjTa1NY to explain the multi-view consistency setup and demonstrate the **Ski-PTZ** and **Handheld190k** video results including the intermediate outputs of our method along with the detection and segmentation predictions on consecutive frames for all cameras and test subjects.

1. Implementation Details

1.1. Multi-view Consistency

Adjusting Bounding Box Centers. The candidate object location proposed by the sampled grid cell in camera c is defined as $\mathbf{b}_c = [\delta x, \delta y, s_x, s_y]$ where $\delta x, \delta y \in [0,1]$ are the offsets from the grid center and $s_x, s_y \in [0,1]$ are the width and height of the bounding box respectively. The center location of the proposal in pixel coordinates can be written as

$$\begin{aligned} u_c &= W * \delta x + g_x \ , \\ v_c &= H * \delta y + g_y \ , \end{aligned} \tag{1}$$

where $u_c \in [0, W]$, $v_c \in [0, H]$ and $g_x \in [0, W]$, $g_y \in [0, H]$ denote the grid center in pixel coordinates. To reach a multi-view consensus on the center of a 3D bounding box, namely $\bar{\mathbf{u}} \in \mathbb{R}^{3\times 1}$, we take into account the lines emerging from camera positions $\mathbf{o}_c \in \mathbb{R}^{3\times 1}$ for each camera. The line of sight for the proposal center is calculated as

$$\mathbf{l}_c = \mathbf{M}_c^{-1} \begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \ , \tag{2}$$

where $\mathbf{l}_c$ represents all the points corresponding to the center of the sampled box in world coordinates relative to the camera center and $\mathbf{M}_c$ is the 3×3 matrix formed by the first 3 columns of the projection matrix $\mathbf{P}_c$. Note that we use bold symbols ($\mathbf{l}_c$) for vectors in 3D world space and normal letters (u and v) for coordinates in the 2D image plane. The unit direction vector for each of these lines is

$$\mathbf{n}_c = \frac{\mathbf{l}_c}{\|\mathbf{l}_c\|} \ , \tag{3}$$

where $\mathbf{n}_c \in \mathbb{R}^{3\times 1}$. To find the nearest point $\bar{\mathbf{u}}$ to a set of lines, we calculate the point with minimum distance to them. Given that each line is defined by its origin $\mathbf{o}_c$ and the unit direction vector $\mathbf{n}_c$, the squared perpendicular distance from the point $\bar{\mathbf{u}}$ to one of these lines is given by

$$\mathbf{d}_c = (\mathbf{o}_c - \bar{\mathbf{u}})^T (\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T})(\mathbf{o_c} - \bar{\mathbf{u}}) \ , \tag{4}$$

where the matrix $(\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T})$ serves as the projector of the line vectors into the space orthogonal to $\mathbf{n}_c$. By minimizing the sum of squared distances, we can obtain the nearest point in the least squares sense for C cameras. The objective we want to minimize is

$$\sum_{c=1}^{C} \mathbf{d}_c = \sum_{c=1}^{C} (\mathbf{o}_c - \bar{\mathbf{u}})^T (\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T})(\mathbf{o_c} - \bar{\mathbf{u}}) \ . \tag{5}$$

The derivative with respect to $\bar{\mathbf{u}}$ gives

$$\sum_{c=1}^{C} -2(\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T})(\mathbf{o_c} - \bar{\mathbf{u}}) = \mathbf{0} \ , \tag{6}$$

where $\mathbf{I}$ is the 3×3 identity matrix. Re-arranging this, we obtain a system of linear equations

$$\begin{aligned} \mathbf{A}\bar{\mathbf{u}} &= \mathbf{m} \ , \\ \mathbf{A} &= \sum_{c=1}^{C} (\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T}) \ , \\ \mathbf{m} &= \sum_{c=1}^{C} (\mathbf{I} - \mathbf{n_c}(\mathbf{n_c})^\mathbf{T})\mathbf{o_c} \ , \end{aligned} \tag{7}$$

with $\mathbf{A} \in \mathbb{R}^{\mathbf{3\times3}}$ and $\mathbf{m} \in \mathbb{R}^{\mathbf{3\times1}}$. The optimum is achieved at the least squares solution. Therefore, $\bar{\mathbf{u}} = lstsq(\mathbf{A}, \mathbf{m})$ and we use a differentiable implementation of *lstsq* function to solve it.

The new center computed through multi-view consistency is projected onto each view to update the value of the 2D bounding box centers. Thus,

$$\begin{bmatrix} \bar{u}_c \\ \bar{v}_c \\ 1 \end{bmatrix} = \mathbf{M}_c \bar{\mathbf{u}} \ , \tag{8}$$

where $\bar{\mathbf{u}}$ represents the coordinates of the new center in 3D and $\{\bar{u}_c, \bar{v}_c\}$ are the updated 2D bounding box centers in each view.

Adjusting Bounding Box Heights. Similarly, the top and bottom points of the 2D bounding boxes can be subject to the multi-view consistency. The top and bottom locations, $\{u_{t,c}, v_{t,c}\}$ and $\{u_{b,c}, v_{b,c}\}$ respectively, of the bounding box in camera view c are computed as

$$\begin{aligned} u_{t,c} &= W * \delta x + g_x \ , \\ v_{t,c} &= H * \delta y + g_y - (H * s_x)/2 \ , \\ u_{b,c} &= W * \delta x + g_x \ , \\ v_{b,c} &= H * \delta y + g_y + (H * s_x)/2 \ . \end{aligned} \tag{9}$$

To find the consensus top and bottom locations in 3D, we apply the least squares solution explained in the previous section separately to the top and bottom points. For the top point, we consider the set of lines originating from camera positions $\mathbf{o}_c \in \mathbb{R}^{3\times1}$ for each camera. The line of sight for the top and bottom points of the bounding box in camera view c are given as

$$\begin{aligned} \mathbf{l}_{t,c} &= \mathbf{M}_c^{-1} \begin{bmatrix} u_{t,c} \\ v_{t,c} \\ 1 \end{bmatrix} \ , \\ \mathbf{l}_{b,c} &= \mathbf{M}_c^{-1} \begin{bmatrix} u_{b,c} \\ v_{b,c} \\ 1 \end{bmatrix} \ . \end{aligned} \tag{10}$$

To find the nearest point $\bar{\mathbf{u}}_t$ to these lines, we apply Eq. 3, 4, 5, 6 and 7. Finally, we obtain the updated pixel location for the top point of the bounding box as follows

$$\begin{bmatrix} \bar{u}_{t,c} \\ \bar{v}_{t,c} \\ 1 \end{bmatrix} = \mathbf{M}_c \bar{\mathbf{u}}_t \ . \tag{11}$$

We update the 2D bottom location using the same multi-view least-squares strategy.

1.2. Architectures

Our main network $\mathcal{F}$ consists of a detection and a synthesis network that reconstruct the input scene against the background image generated by the inpainting network.

Detection network. We predict one candidate bounding box relative to each 2D grid cell in a regular 8×8 grid using a fully-convolutional architecture similar to YOLO [4]. We use a ResNet-18 backbone [1] without pre-training, that reduces the input dimensionality by a factor 16, forming a low resolution grid of features, e.g., to spatial resolution 8×8 from 128×128. The feature size is set to 5; two for bounding box location offset $\delta x, \delta y \in [0, 1]$, two for scale $s_x, s_y \in [0, 1]$, and one for the probability p. Each feature output represents the bounding box parameters predicted by one grid cell, and the offset is relative to the cell center $\{g_x, g_y\}$. The output p is forced to be positive and form a proper distribution, with $\sum_{i=1}^{N} p_i = 1$ where $N = 64$, by a soft-max activation unit. To prevent this network from constantly predicting bounding boxes at the borders of the image, we zero out the outer cell probabilities.

Synthesis network. This network takes as input the cropped image region corresponding to the sampled bounding box and has the form of a bottle-neck auto-encoder, based on the publicly available implementation of [6]. The encoding part is a 50-layer residual network, and the weights are initialized with ones trained on ImageNet classification. The hidden layer is 856 dimensional, split into a 600 dimensional space and a 256 dimensional space that is replicated spatially to a $512 \times 8 \times 8$ feature map to encode spatially invariant features. The decoding is done with the second half of a U-Net architecture with 64, 128, 256, 512 feature channels in each stage. The final network layer outputs four feature maps, three to predict the color image $\hat{\mathbf{I}} \in \mathbb{R}^{128\times128\times3}$ and one for the segmentation mask $\mathbf{S} \in \mathbb{R}^{128\times128}$.

Inpainting network. The inpainting network is trained separately for each dataset, from scratch and on the training split, without requiring any annotation. It is a 6 layer U-Net model with 8, 16, 32, 64, 128, 256 feature channels in each stage. It is trained independently from the rest of the pipeline by feeding images with randomly occluded regions of varying sizes. To compare the reconstructed image $\mathbf{I}'$ to the original one $\mathbf{I}$, we use the L_2 pixel reconstruction and perceptual losses

$$L_{reconst} = ||\mathbf{I} - \mathbf{I}'||^2 \ , \tag{12}$$

$$L_{perc} = ||\phi(\mathbf{I}) - \phi(\mathbf{I}')||^2 \ , \tag{13}$$

where $\phi(.)$ indicates the low level features obtained by passing its input to a pre-trained ResNet18 network. The pixel reconstruction and perceptual losses are weighted 1:2.

We integrate the inpainting network to our full pipeline and use it in an off-the-shelf manner. The input to the inpainter is an image where the selected bounding box region is hidden and the output is the entire image with the initially hidden patch being reconstructed. In our full pipeline, the weights of the inpainting network are frozen and to remove the image evidence corresponding to the foreground person, the hidden patch in the input image to the inpainting network is selected to be the bounding box region expanded by 15% in both dimensions.

1.3. Training Details

Overall training. We train our model with L_2 pixel reconstruction and perceptual losses on the reconstructed image $\mathcal{F}(\mathbf{I}_c)$ and the L_2 pixel reconstruction loss on the inpainted background image $\bar{\mathbf{I}}_c$ in view c . We rely on the same prior terms as in [2] to regularize the predicted segmentation masks and probability values for the voxels. We use an L_{seg} prior, which encourages the mean value of a segmentation mask to be larger than a threshold lambda but small in general,

$$L_{seg} = \left| \left(\frac{1}{WH} \sum_{x}^{W} \sum_{y}^{H} \mathcal{T}^{-1}(\mathbf{S})_{xy} \right) - \lambda \right| + \lambda \;, \tag{14}$$

where λ is set to 0.005 and the notation is the same as in the main paper. It encourages a non-zero segmentation mask at the beginning of the training, when the decoder still produces non-perfect foreground, which improves and stabilizes convergence. The voxel probabilities q_j are regularized with

$$L_q = \sum_{j}^{V} |q_j| \tag{15}$$

that favors only few voxels to have non-zero values. The total training loss we minimize can be written as

$$\begin{aligned} L_{total} = &- \alpha \sum_{c=1}^{C} r_j \frac{\|\bar{\mathbf{I}}_c - \mathbf{I}_c\|^2}{area(\mathbf{b}^c_{i^c(j)})} \\ &+ \beta \sum_{c=1}^{C} r_j \|\mathcal{F}(\mathbf{I}_c) - \mathbf{I}_c\|^2 \\ &+ \gamma \sum_{c=1}^{C} r_j \|\phi(\mathcal{F}(\mathbf{I}_c)) - \phi(\mathbf{I}_c)\|^2 \\ &+ \eta \sum_{c=1}^{C} L^c_{seg} + \zeta L_q \end{aligned} \tag{16}$$

where $\alpha = 0.1, \beta = 1, \gamma = 2, \eta = 0.25, \zeta = 0.1$ and $\phi(.)$ indicates the low level features obtained by passing its input to a pre-trained ResNet18 network. The first three terms of L_{total} correspond to $G(\mathbf{I}_1, \ldots, \mathbf{I}_C)$, $O(\mathbf{I}_1, \ldots, \mathbf{I}_C)$ and the perceptual version of $O(\mathbf{I}_1, \ldots, \mathbf{I}_C)$ defined in Section 3.1.4 of the main document.

As a baseline (*Ours w/ TC*), we report the results of using a L_2 loss term to minimize the distance between lines passing through the initial bounding box centers and camera optical centers in each view.

All training stages are performed on a single NVIDIA TITAN X Pascal GPU with Adam and a learning rate of 1e-3. First, the inpainting network is optimized for 100k iterations and subsequently the complete network for an additional 50k iterations. The decoding part of the synthesis network uses a reduced learning rate of 1e-4, to prevent occasional diverging behavior. We use a batch size of 48 and an input image resolution of 640px$\times$360px for the **Ski-PTZ** and **Handheld190k** and 500px$\times$500px for **H36M**.

Importance sampling. Sampling from a discrete distribution is not differentiable with respect to its parameters. Therefore, we integrate importance sampling as in [2]. However, instead of sampling from a 2D grid of cells, we sample a voxel from a 3D grid of proposals. Importance sampling allows us to introduce an auxiliary distribution k that is used as the importance sampling distribution while maintaining the differentiability and optimizing the voxel probability distribution q. The relationship between k and q can be expressed as

$$k_j = q_j(1 - V\epsilon) + \epsilon \tag{17}$$

for a voxel j, where $(1 - V\epsilon)$ determines the probability of choosing a random voxel. In the multi-view setting, the number of voxels that can be seen by all the cameras change from one frame to another. Therefore the importance sampling related hyper-parameters must be adjusted accordingly. As in [2], we take $(1 - V\epsilon) = 0.064$ and to satisfy this equality, we use an adaptive $\epsilon \approx 0.0002$, which makes the method numerically stable while the probability of choosing a random bounding box stays low, i.e., 6.4% for on average $V = 300$ voxels that participate the multi-view consensus voting. In Section 3.1.4 of the main document, r_j is the ratio of the probability q_j by its importance sampling probability k_j.

Consistency. We demonstrate that the proposed training strategy is stable and produces consistent results when repeated using the same configuration. To this end, we train the best-performing model on the **Ski-PTZ** and **H36M** datasets three times from scratch and provide the mean and std of the scores on the test sequences. The J- and F- measures on the **Ski-PTZ** dataset are consistent, respectively, 0.71 ± 0.006, 0.83 ± 0.002 and the mAP$_{0.5}$ score on **H36M** dataset is 0.85 ± 0.004.

2. Qualitative Results

We present additional qualitative results on **Ski-PTZ**, **H36M** and **Handheld190k** in Fig. 1, Fig. 2 and Fig. 3 respectively. On **Ski-PTZ**, our method reliably detects the skier even when there are other people in the scene and our segmentation predictions cover the entire body and skis more accurately than [7], relying on multi-view consistency

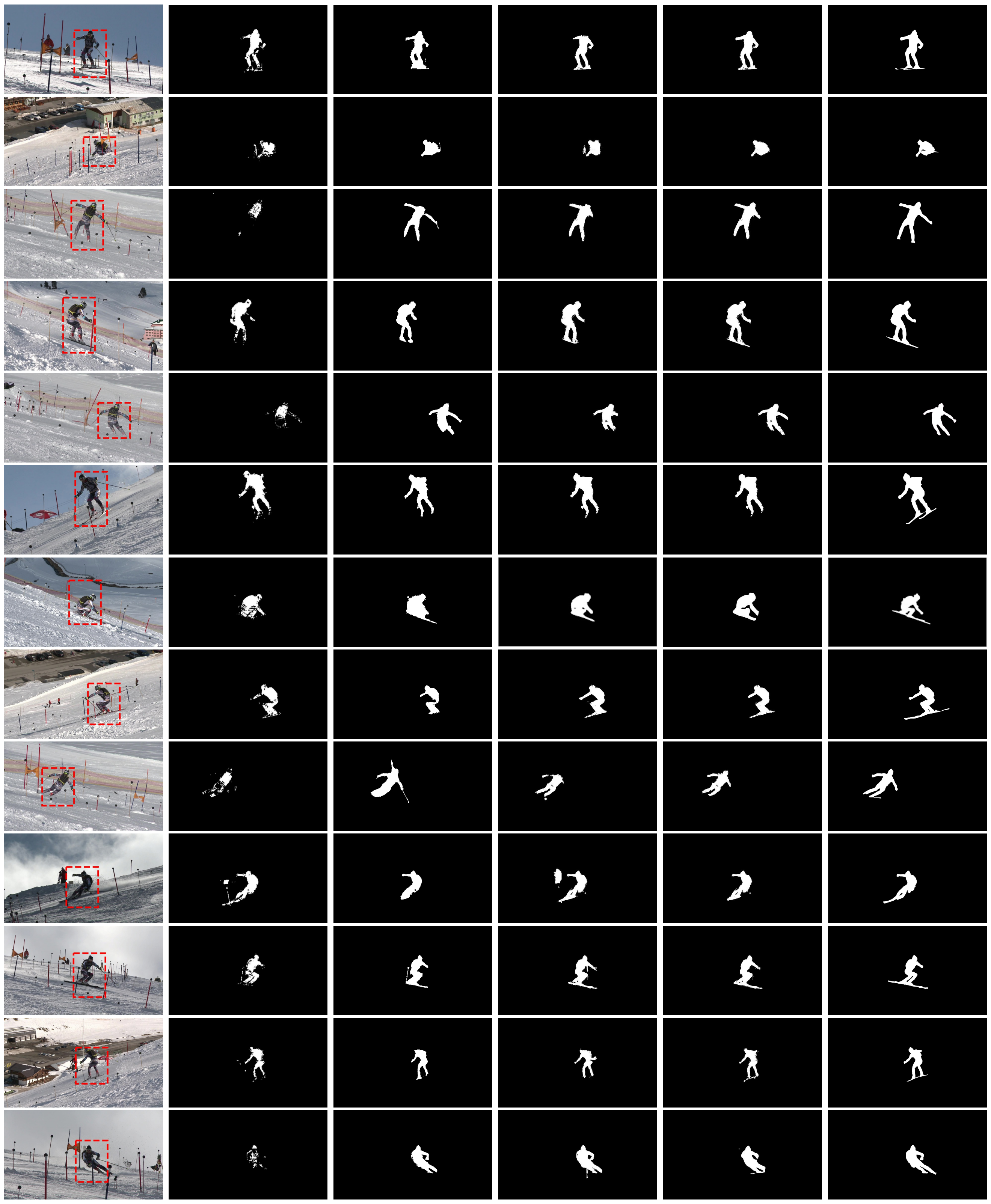
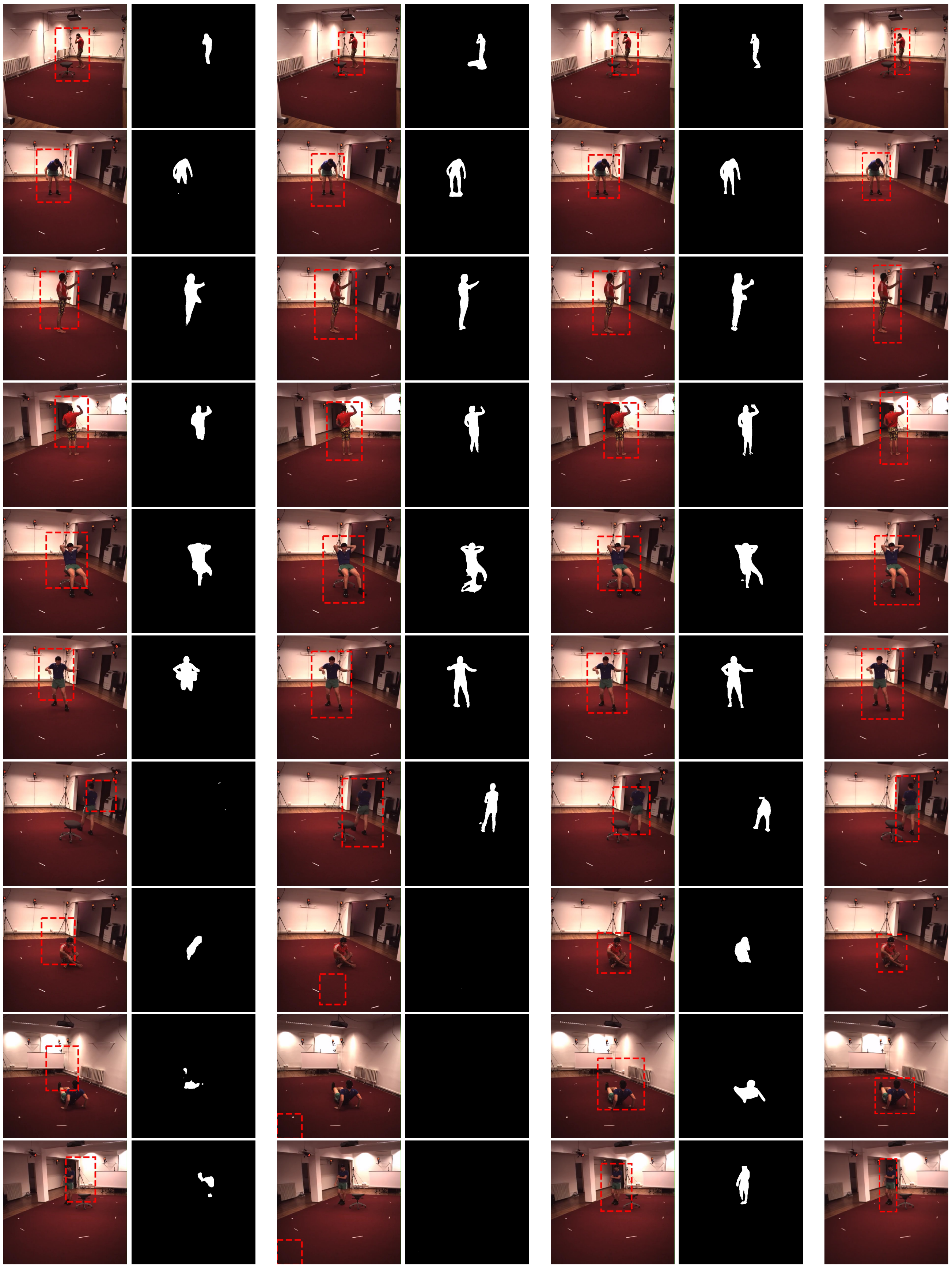
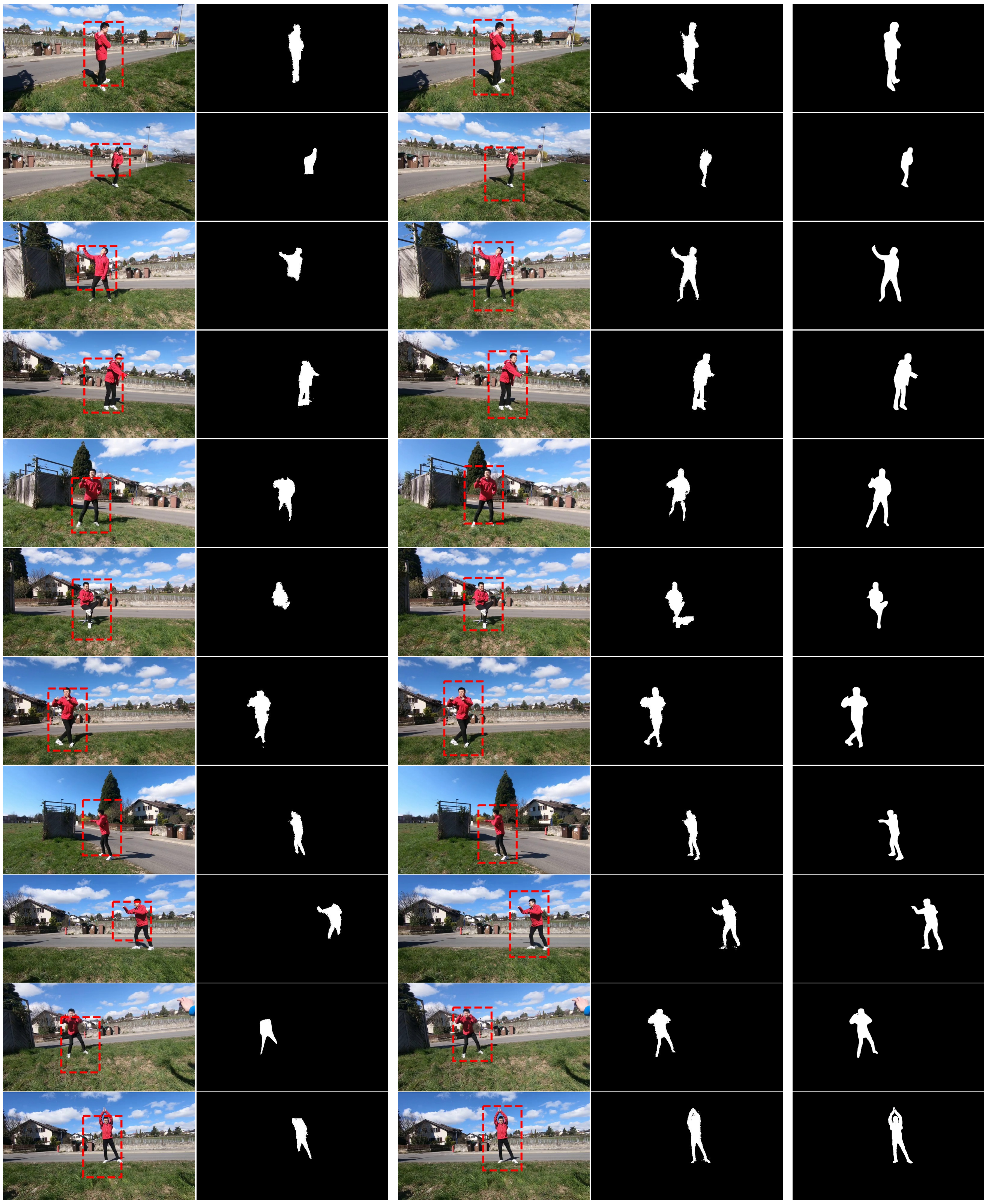


Figure 1. **Qualitative results on the Ski-PTZ.** (a) Input images with our predicted bounding box overlaid in red. (b) Segmentation mask prediction of [7]. (c) Segmentation mask prediction of [3]. (d) Segmentation mask prediction of [2] (e) Our segmentation mask prediction. (f) Ground truth segmentation mask. Note that, unlike our method, [3] and [7] use explicit temporal cues at inference time.



(a) Katircioglu et al. [2] (b) Rhodin et al. [5] (c) Ours (d) GT

Figure 2. **Qualitative results on the H36M dataset.** (a) The detection and segmentation mask results of Katircioglu et al. [2] trained and tested on single images. (b) The results of [5] trained with a pair of camera views and tested on single images. (c) Our predictions obtained from the model trained with the 4-cam multi-view consistency and tested on single images. (d) Ground truth.



(a) Katircioglu et al. [2] (b) Ours (c) GT

Figure 3. **Qualitative results on the Handheld190k dataset.** (a) The detection and segmentation mask results of [2] trained and tested on single images. (b) The predictions of our model trained using 3-camera multi-view consistency and tested on single images. (c) Ground truth. Our results are generally more accurate, which justifies the effort invested in calibrating the cameras.

during training, whereas [3] uses strong temporal cues both during training and test time and [2] leverages optical flow images during training. Due to the background objective, our approach favors tight bounding boxes around the subject and this causes the auxiliary object moving with the primary one to be partially included in the detection. Therefore, compared to the ground truth masks, our predictions do not contain the skis entirely. However, compared to other baselines, our method can segment out the skis more precisely.

On **H36M** dataset, our method has more accurate hand detections and lower legs are more precisely segmented compared to [2] employing a single view approach with optical flow images during training and [5] using multi-view images during training for novel view synthesis. Even in the rare cases of performing an action on the floor, our method can still reliably detect the person. The failure cases include the detections that miss the head and feet when the chair is in close proximity to the subject. This is expected since the chair is also hard to be reconstructed from its neighboring regions and can be treated as a foreground object.

To demonstrate that our method can be applied to in-the-wild scenes without initial camera calibration, we used the OpenSFM software to calibrate 4200 frames out of 120000 training images in the **Handheld190k** dataset. We ran OpenSFM with HaHOG (the combination of Hessian Affine feature point detector and HOG descriptor) features and the calibration took approximately 7 hours. We did not provide masks for the moving objects. Nonetheless, we managed to obtain accurate camera poses. Our results in Fig. 3 show that when trained with a small calibrated part of the training set, our multi-view approach can detect and segment the person more accurately than [2] which often fails to detect the moving object precisely.

Although we target the detection of a single object or person, our probabilistic framework can handle several of them at test time by sampling more than once. In Fig. 4, we show an example of this on **Ski-PTZ**, by synthetically creating an image with two skiers. Our method trained on single person images can accurately detect and segment two skiers as long as they are sufficiently separated.

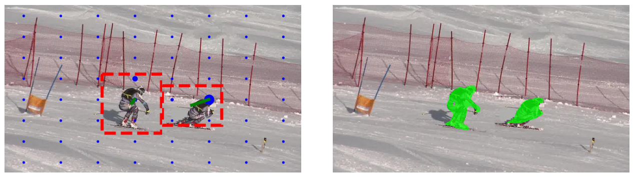

(a) Detection (b) Segmentation

Figure 4. Multi-person detection and segmentation at test time.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[2] I. Katircioglu, H. Rhodin, V. Constantin, J. Spörri, M. Salzmann, and P. Fua. Self-Supervised Segmentation via Background Inpainting. In *arXiv Preprint*, 2020. 3, 4, 5, 6, 7

[3] Y. J. Koh and C.-S. Kim. Primary Object Segmentation in Videos Based on Region Augmentation and Reduction. In *Conference on Computer Vision and Pattern Recognition*, 2017. 4, 7

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition*, 2016. 2

[5] H. Rhodin, V. Constantin, I. Katircioglu, M. Salzmann, and P. Fua. Neural Scene Decomposition for Human Motion Capture. In *Conference on Computer Vision and Pattern Recognition*, 2019. 5, 7

[6] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation. In *European Conference on Computer Vision*, 2018. 2

[7] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto. Unsupervised Moving Object Detection via Contextual Information Separation. In *Conference on Computer Vision and Pattern Recognition*, 2019. 3, 4