Part Affinity Field based Activity Recognition

Thomas Golda

Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
thomas.golda@kit.edu

Technical Report IES-2019-11

Abstract

This report presents work and results on Activity Recognition using Part Affinity Fields for real-time surveillance applications. Starting with a short introduction to the motivation, this report gives a detailed overview over the key idea of the pursued approach and explains the basic ideas. In addition a variety of experiments on various subjects are presented, like i) the impact of the number of input frames, ii) the impact of different simple dimensionality reduction approaches, and iii) a comparison on how multi-class and binary problem formulation influence the performance.

1 Introduction

Anomaly detection amongst other strongly related topics like outlier and novelty detection, plays an important role in various research fields as network traffic monitoring, time series analysis, medical image analysis, and video surveillance. However, when talking about anomalies in the context of video surveillance the understanding of what an anomaly actually is can differ strongly between applications. For instance, an anomaly can be an abandoned suitcase at a public place, a vehicle driving through a pedestrian zone or suspicious or salient behaving people. Recent progress in the fields of classification, pattern recognition and time series prediction has also brought the field of activity recognition into the focus of application-oriented research for surveillance scenarios. This report proposes a new human pose based approach on classifying the behavior of pedestrians. It presents details of the architecture and various experiments on different considered time horizons as well as various considerations that were conducted in order to tackle the problem of activity recognition in such scenarios. The presented approach is preparatory work for future activities on human-centered abnormal behavior detection.

2 Part affinity fields for activity recognition

2.1 Human pose estimation in the wild

Human Pose Estimation describes the problem of estimating a skeletal representation of a person based on information gathered using certain types of sensors. The skeletal representation is typically represented as a graph $G = (V, E)$ where $V \subset \mathbb{R}^n$ is a set of keypoints and $E \subset V \times V$ is a set of edges connecting various keypoints. Depending on the chosen skeletal model the graph can be seen as a tree. Usually the used sensors are classical video cameras or depth cameras delivering RGB or RGB-D information respectively. This work focuses on the 2D case using classical cameras and RGB data. This decision is driven by the corresponding problem domain, namely video surveillance in urban setups, where typical camera setups consist of RGB cameras. To this point, RGB-D cameras are rarely used. As a consequence, the resulting skeletons produced by human pose estimation algorithms consist of keypoints in a two-dimensional space with $V \subset \mathbb{R}^2$.

2.2 Part affinity fields

For this approach, the framework *OpenPose* by Cao et al. [1], which belongs to the group of *bottom-up* methods, is used. Contrary to *top-down* methods, bottom-

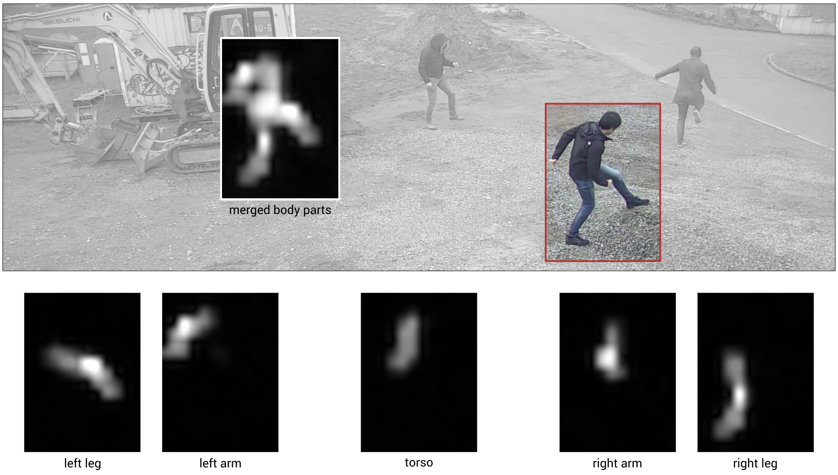

Figure 2.1: Based on the Part Affinity Fields provided by the OpenPose framework body part maps encoding the presence of a body part are constructed. This is done per frame. The resulting body parts $\mathcal{B}$ for the highlighted person are shown in the lower part of the figure. For visualization purposes the body parts are merged into a single layer image to show that they form an understandable representation of pedestrians.

up methods first locate all keypoints in a given image, which are connected in a subsequent step. To do so, the method proposed in [1] computes *Part Affinity Fields* (PAF) that are used to connect estimated keypoints by adding further semantic information about visible body parts. In detail, the computed PAFs are used for constructing a bipartite keypoint graph that is subject to the final optimization problem which is solved using the Hungarian Method [7].

2.3 Architecture

Since the aim of activity recognition in surveillance scenarios is to have a near real-time processing of video footage, typical activity recognition frameworks are not applicable due to their large network architectures and resulting strong hardware requirements. As a result, the focus of this work lies on developing an approach using a much smaller neural network. For the task of image classification

Howard et al. [3] proposed a network architecture called MobileNet that is much smaller, i.e. much less parameters, than most competing architectures achieving comparable performance at the same time. Due to the promising performance of MobileNet for the classification of single images the decision was made to adapt it for an custom and fast activity recognition approach. This choice brings a constraint into the considerations: Working on raw 2D keypoint coordinates is not possible with the pre-defined architecture. However, two possibilities come up when dealing with this problem. The first way is to transform the keypoint coordinates into images, the second to use the body representation already provided by the OpenPose framework. Since it is easy to obtain the latter and is available out of the box when using OpenPose the decision was made to adapt the Part Affinity Fields instead of the raw coordinates.

2.3.1 From part affinity fields to human body parts

In order to reduce the number of inputs semantically corresponding Part Affinity Fields $F_{\text{part}} = (F_{\text{part},x}, F_{\text{part},y})$ are aggregated to five body parts: torso, left arm, right arm, left leg and right leg. The following equation shows the formula for computing the corresponding body part using the Part Affinity Fields related to the left leg.

$$\mathcal{B}_{\text{leftLeg}} = \sqrt{\lambda \cdot (\mathcal{F}^2_{\text{leftCalf,x}} + \mathcal{F}^2_{\text{leftCalf,y}})} + \sqrt{\lambda \cdot (\mathcal{F}^2_{\text{leftThigh,x}} + \mathcal{F}^2_{\text{leftThigh,y}})} \quad (2.1)$$

where $\lambda \in \mathbb{R}^+$ is a scaling factor. Note that $\mathcal{B}$ encodes the presence of a body part rather than its direction since this information is lost by transforming the Part Affinity Fields into body parts. However, since the information about single body parts is still available and no further reducing operations are performed, it is still possible to infer the orientation of the represented person.

2.4 Training dataset

The decision to use Part Affinity Fields as input to the model architecture makes it impossible to use a pre-trained network since the input volume has five instead

of three channels. Furthermore, the input channels do not correspond to typical structures like they can be encountered in RGB images. Therefore the chosen architecture has to be trained from scratch.
In order to train the network three existing datasets were merged: INRIA Xmas Motion Acquisition Sequences (IXMAS) [11], UT-Interaction [10] and IOSB Multispectral Action Dataset [2]. All these datasets come with different sets of activities A_{IXMAS}, $A_{\text{UT-Interaction}}$ and A_{IOSB}. For this work, the available activities were merged to get a total of 19 activities from all three datasets. About 80% of the available activities come from IXMAS, which is the most diverse dataset of these three. Another reason to chose these datasets is motivated by the viewing perspective and the size of the urban outdoor environment, which showed the best fit to the field of application.
Since all datasets consist of video sequences it is possible to make use of temporal information, which would be typical done by tracking pedestrians. For the initial setup no tracking is considered for performance reasons as tracking of multiple targets would introduce further expensive computations. However, an alternative way to benefit of the available temporal information is inspired by the *anchor cuboids* used in [4]. A schematic overview over this principle is given in Figure 2.2. Given a bounding box B_t at timestep t and a window size k an input volume is constructed by simply aggregating the spatial information at the location of B_t over the last k timesteps

$$\tilde{B}_{t,k} = B_{t-k+1} \oplus ... \oplus B_{t-2} \oplus B_{t-1} \oplus B_t \tag{2.2}$$

where $\oplus$ describes the concatenation operation along the channel axis. Note that each bounding box B_i contains spatially corresponding information from all five body part channels and hence can be written as

$$B_i = (\mathcal{B}_{\text{leftLeg}}, \mathcal{B}_{\text{leftArm}}, \mathcal{B}_{\text{torso}}, \mathcal{B}_{\text{rightArm}}, \mathcal{B}_{\text{rightLeg}}) \tag{2.3}$$

2.4.1 Multi-class approach

As mentioned in the introductory part of this section the used dataset consists of 19 activity classes with sequences taken from three different public datasets. The

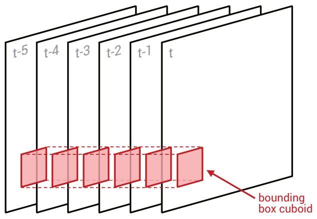

Figure 2.2: In order to avoid further computation overhead by performing tracking of pedestrians, temporal cuboids are build as shown in red. For a given timestamp t, a given window size k and a bounding box B_t we merge the content of the corresponding bounding boxes B_{t-k+1}, ..., B_t. All bounding boxes share the same location at different points in time.

set of available activities ranges from everyday activities like *walking*, *sitting down* up to more unusual ones like *kicking* and *punching*. In the multi-class approach every movement of a pedestrian is classified into one of the available classes. For the training of the network the broadly utilized Adam optimizer [5] was used with an initial learning rate of 10^{-3}. As the training objective, cross-entropy was chosen as it is the most common loss for classification tasks. The learning rate is reduced by a factor of 0.1 after each 20 epochs without improvement on the used validation set. The whole training was conducted on an Nvidia DGX-1 using a single Tesla V100 card with 32 GiB of memory. This allows to use large batches with around 500 samples per batch. The actual number of samples contained in a single batch is chosen empirically and depends on the regarded number of timesteps k. Figure 2.3 illustrates the overall setup of the final architecture, which takes as input a set of k subsequent body part sets of a given person detection $\tilde{B}_{t,k}$. The input is then processed by the adapted MobileNet model and classified as one of the 19 regarded activity classes.

2.4.2 Binary approach

In addition to the multi-class approach, a binary classification task with the aim to distinguish between target activities and non-target activities was investigate. As target activities a subset of activities, namely *kick*, *punch*, *hit* and *push* were chosen, since they show quite similar and relevant activities. To encode these

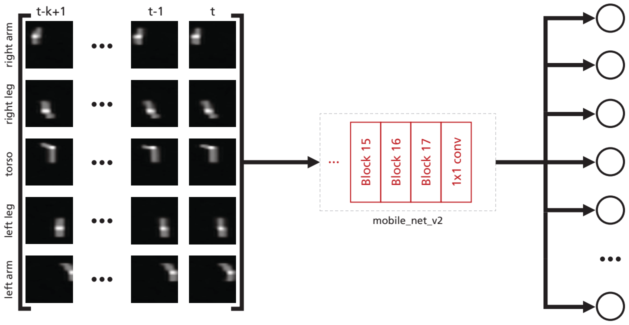

Figure 2.3: Input to the model is a volume of size $224 \times 224 \times 5k$ where k is the number of timesteps that are taken into account. Body parts are stacked along the third axis over time. Each detected pedestrian and its corresponding body part maps are resized to 224×224 as expected by the MobileNet [3] architecture. Every row on the left corresponds to one of the five body parts $\mathcal{B}_i$, each column to a considered timestep. The number of neurons in the final classification layer is furthermore changed to the number of activity classes $c \in \{2, 19\}$.

two classes, the number of output neurons was reduced to two neurons in the model architecture.

A consequence of transforming the multi-class to a binary classification problem is the accentuation of the imbalance between the regarded classes. To address this problem, the imbalance was considered implicitly by changing the used training loss. For this reason the Focal Loss [8] was adapted, which is an extension of the classical cross-entropy loss that introduces a weighting of samples based on the quality of their already achieved classification result.

$$\mathcal{L}_{\text{focal}}(p_t) = -(1 - p_t)^{\gamma} \cdot \log(p_t) \qquad \text{with} \quad \gamma \in \mathbb{R}_0^+ \tag{2.4}$$

As can be seen in the equation above, the difference to the cross-entropy loss comes from an additional factor $(1 - p_t)^{\gamma}$ that reduces the loss for well-classified samples ($p_t > 0.5$). The introduced variable $\gamma \in \mathbb{R}_0^+$ controls how strong the influence of the well-classified samples to the overall loss can get. The higher the value of γ the more the samples on which the regarded model already achieves good results affect the computed gradients and hence the training process.

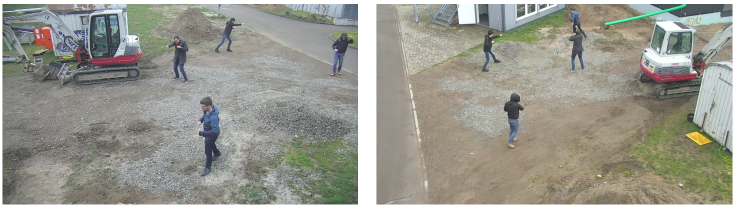

Figure 3.1: In order to evaluate the impact of the presented considerations on the overall performance of the approach a dataset was created using two cameras mounted in different heights pointing to the same location that is only used for evaluation.

3 Evaluation

3.1 IOSB-Ka dataset

For the evaluation an eligible dataset was created that shares many properties with typical public surveillance scenarios. The dataset was recorded at the Fraunhofer Institute for Optronics, System Technologies and Image Exploitation IOSB in Karlsruhe using a common video surveillance setup. It consists of six sequences with an average duration of 56 seconds from two different cameras both showing the same location. The cameras were mounted with different orientations and at different heights. Each video sequence shows a group of people performing actions from a predefined action set that comprises actions like *kicking*, *punching* and *waving*. Figure 3.1 shows two randomly selected frames each taken from one of the two cameras.

3.2 Temporal window size

The first part of our experiments were conducted to examine the influence of the temporal window on the overall performance. Since all sequences from our training dataset were recorded with frame rates between 25 and 30 frames per second, the number of consecutive regarded frames has to be chosen long enough to capture the important part of an action. Therefore a series of experiments was performed for values of $k \in \{1, 6, 10, 14, 18, 24\}$. As stated earlier the input

Table 3.1: The average precision on the evaluation dataset for the provided activities and the overall mean average precision vary slightly for different values of k. The corresponding values are indicated by the identifier PBAR-MFk. The model without temporal information, i.e. $k = 1$, is referred to as PBAR-SF. The best results were achieved with a windows size of $k = 10$.

	kick	**punch**	**wave**	**mAP**
PBAR-SF	0.32	0.37	0.42	0.37
PBAR-MF6	0.34	0.37	0.51	0.40
PBAR-MF10	$\mathbf{0.35}$	0.42	$\mathbf{0.66}$	$\mathbf{0.48}$
PBAR-MF14	0.32	0.39	0.52	0.41
PBAR-MF18	0.33	0.40	0.64	0.46
PBAR-MF24	0.30	$\mathbf{0.45}$	0.59	0.45

to the model is a sequence of k consecutive 5-tuples and can be seen as five sequences showing the temporal behavior of different body parts. Table 3.1 shows the results for different window sizes. It is obvious that all approaches perform almost identical for both activities *kick* and *punch*. Even for a time window of almost a second ($k = 24$) the results do not improve. However, the results for *wave* are better and the benefit of including temporal information is clearly visible. The reason for the results on the first two activities might be due to very similar motions in the training dataset and the far wider variety of forms for the same activity in the test set. Another explanation could be, that the model could not learn to distinguish between similar activities. This has to be investigated in future with additional experiments.

3.3 Dimensionality reduction

A major drawback that comes up when increasing the number of timesteps and hence the size of the input volume to the neural network is the rising computational complexity at training as well as at inference time. While the first is not too much of a problem, the latter means a direct effect on the frame rate and hence on the ability to be close to real-time. As a consequence the question comes up, whether a reduction of dimensionality achieves comparable performance

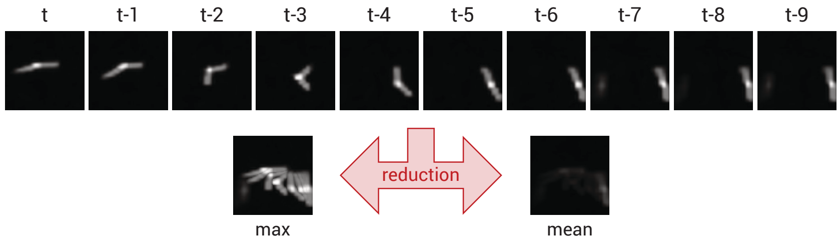

Figure 3.2: Given a timestep t and a corresponding input volume $\tilde{B}_{t,k}$, each body part of the input volume is reduced to a single channel. The illustration shows the result of the reduction exemplary for the right leg $\mathcal{B}^{i}_{\text{rightLeg}}$, where i stands for the corresponding offset to the current timestep. Given k timesteps, the chosen reduction function $f_{\text{reduce}}(\{\mathcal{B}^{i}_{\text{rightLeg}} \mid i \in \{0, \dots, k-1\}\})$ is applied merging on each pixel along the time axis. It is obvious that the *max*-reduction is much more comprehensive for the human, however, the *mean*-reduction does not use any relevant information. It keeps information about the velocity of the action through the amplitude of the output signal.

to the full temporal information. Hence, the network was trained on merged inputs using two ways to reduce the dimensionality: *max* and *mean* reduction. Furthermore, the decision was made to keep the spatial information and perform reduction just over time dimension, i.e. fusing just the information corresponding to the same body part. Figure 3.2 illustrates how the dimensionality reduction works in principle and shows the corresponding outcome for our two considered reduction functions.

Table 3.2: In order to investigate the effect of dimensionality reduction two simple approaches were applied to the best performing model PBAR-MF10: *max* and *mean* reduction. In both cases the resulting reduced input volumes do not carry enough information, so that the performance of the trained model drops significantly. The last row also provides results for the non-reduced model as comparison.

	kick	**punch**	**wave**	**mAP**
max	0.29	0.29	0.57	0.38
mean	0.28	0.33	0.57	0.40
without	**0.35**	**0.42**	**0.66**	**0.48**

Applying reduction to the input decreases training time approximately about 80% from 29.5 minutes to 5.7 minutes per epoch. This effect cannot be observed during inference time where the overall processing time stays identical. A possible reason for this might be that the training is performed using Python. Python is not optimized for memory efficiency and hence moving data from RAM to VRAM might be a bottleneck. The final productive system is written in C++ using the libtorch library provided by the PyTorch development team, which seems to work more efficient when shifting data between devices. However, Table 3.2 indicates that in both cases the performance decreases significantly by a similar amount when using these simple reduction mechanisms.

Since *mean* and *max* reduction appeared to be too strict approaches according to the results presented in Table 3.2, further investigations on the impact of dropping intermediate frames in order to reduce the input size were performed. The dropping is performed in an equally spaced manner using an offset $s \in \mathbb{N}$ and hence results in timesteps $t, t-s, t-2s, ..., t-(k-1)\cdot s$. Written in a more compact way, a sequence of k timesteps with an offset s consists of ordered samples B_{t-i} with $i \in I_{k,s}$ and

$$I_{k,s} = \{\, i \in \mathbb{N}_0 \mid i \leq (k-1)\cdot s \ \wedge\ \exists\, m \in \mathbb{N}_0 : i = m \cdot s \,\} \tag{3.1}$$

This method is referred to as *striding*. Furthermore, for given $k, s \in \mathbb{N}$ the function $\kappa(k, s)$ describes the *sampling window size*.

$$\kappa(k, s) = (k-1)\cdot s + 1 \quad \forall\, k, s \in \mathbb{N} \tag{3.2}$$

For $s = 1$ this resembles the original set of timesteps $t, ..., t-k+1$ for a given *input window size* $k \in \mathbb{N}$. Hence, the sampling window size κ equals the input window size k. Table 3.3 shows results for a stride $s = 3$ on input window sizes of $k \in \{6, 10\}$ and compares them to results of approaches with a similar sampling window size without striding. The decision to use a stride of 3 was made empirically. The results indicate that the performance is almost identical to the non-strided experiments and therefore it is possible to achieve similar performing results on a reduced frame rate. By using a stride $s = 3$ the effective

Table 3.3: Two strided approaches with an offset $s = 3$ and *input window size* of $k \in \{6, 10\}$ are compared with non-strided that share a similar sampling window size κ. The results show that the sampling window size is more important than the number of actual considered frames, since the performances does not decrease significantly when taking less frames for comparable κ.

	κ	**kick**	**punch**	**wave**	**mAP**
PBAR-MF18	18	**0.33**	0.40	0.64	**0.46**
PBAR-MF6s3	16	0.32	0.44	0.54	0.43
PBAR-MF24	24	0.30	**0.45**	0.59	0.45
PBAR-MF10s3	28	0.28	0.41	**0.65**	0.45

frame rate is approximately one third of the original and hence around 10 frames per second.

3.4 Multi-class vs. binary problem formulation

For the evaluation of the binary problem formulation PBAR-MF10 was again chosen as baseline. As explained in Section 2.4.2, the number of classes was reduced. The idea behind this decision was that it might be easier for the network to distinguish between ordinary and non-ordinary activities. Splitting the available data in such manner would lead to more samples per class and hopefully a better performance. As Table 3.4 indicates this is the case. By

Table 3.4: For this experiment again the best performing architecture, PBAR-MF10, was chosen and trained in binary and multi-class manner. On the presented test set the binary approach performs approximately 18% mAP better than its multi-class counterpart.

	mAP
multi-class	0.48
binary	**0.66**

tackling the problem using a binary problem formulation the mean average precision could be increased on the regarded test set by about 18% mAP.

However, a major drawback of the binary formulation is that it loses the ability to distinguish between the kind of activity that was perceived. This makes it more difficult to understand the decision of the model, especially when the target class consists of a variety of different activities.

4 Conclusion

This report presents our work on Part Affinity Field based Activity Recognition. It gives an overview over how to include the information of the Part Affinity Fields provided by the OpenPose framework into a lightweight approach, which is designed to work for real-time applications. Furthermore, various topics like i) the impact of the number of input frames, ii) the impact of different simple dimensionality reduction approaches, and iii) a comparison between multi-class and binary problem formulation and how they influence the performance were evaluated. Future work will address further aspects in order to improve the performance and take a closer look on the temporal aspect of the approach: Does the usage of tracking algorithms improve the performance compared to the temporal cuboid approach? Can we benefit from the incorporation of Spatio-Temporal Affinity Fields [9]? How does MobileNet with 3D convolutions [6] perform? In addition to that, more elaborate yet fast dimensionality reduction approaches like PCA or LDA as well as incorporating an understanding of similarity of activities into the approach will be subject to future investigations.

References

[1] Zhe Cao et al. “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *CoRR* abs/1611.08050 (2016). arXiv: `1611.08050`. URL: `http://arxiv.org/abs/1611.08050`.

[2] Barbara Hilsenbeck et al. “Action Recognition in the Longwave Infrared and the Visible Spectrum Using Hough Forests”. In: *IEEE International Symposium on Multimedia, ISM 2016, San Jose, CA, USA, December 11-13, 2016*. 2016, pp. 329–332. DOI: `10.1109/ISM.2016.0072`. URL: `https://doi.org/10.1109/ISM.2016.0072`.

[3] Andrew G. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: *CoRR* abs/1704.04861 (2017). arXiv: `1704.04861`. URL: `http://arxiv.org/abs/1704.04861`.

[4] Vicky Kalogeiton et al. “Action Tubelet Detector for Spatio-Temporal Action Localization”. In: *ICCV 2017 - IEEE International Conference on Computer Vision*. Venice, Italy, Oct. 2017.

[5] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: `http://arxiv.org/abs/1412.6980`.

[6] Okan Köpüklü et al. “Resource Efficient 3D Convolutional Neural Networks”. In: *CoRR* abs/1904.02422 (2019). arXiv: `1904.02422`. URL: `http://arxiv.org/abs/1904.02422`.

[7] H. W. Kuhn and Bryn Yaw. “The Hungarian method for the assignment problem”. In: *Naval Res. Logist. Quart* (1955), pp. 83–97.

[8] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2999–3007. DOI: `10.1109/ICCV.2017.324`. URL: `https://doi.org/10.1109/ICCV.2017.324`.

[9] Yaadhav Raaj et al. “Efficient Online Multi-Person 2D Pose Tracking with Recurrent Spatio-Temporal Affinity Fields”. In: *CoRR* abs/1811.11975 (2018). arXiv: `1811.11975`. URL: `http://arxiv.org/abs/1811.11975`.

[10] M. S. Ryoo and J. K. Aggarwal. *UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)*. 2010.

[11] Daniel Weinland, Rémi Ronfard, and Edmond Boyer. "Free viewpoint action recognition using motion history volumes". In: *Computer Vision and Image Understanding* 104.2-3 (2006), pp. 249–257. DOI: `10.1016/j.cviu.2006.07.013`. URL: `https://doi.org/10.1016/j.cviu.2006.07.013`.