This appendix is organized as follows:

7. Method details

7.1. Sinkhorn optimal transport

To acquire the optimal assignment from word features to video frames, an assignment matrix $\mathbf{Q}$ is computed from each video and ASR pair as shown in Figure 2(a). This cross-model optimal transport mechanism is applied to assignment $\mathbf{Q}$ from the projected cross-model similarity $\mathbf{P}$ between word tokens and each video frame, where $\mathbf{P} = g(\mathcal{S}) \bigotimes f(\mathcal{V})^\top \in \mathbb{R}^{K \times U}$. To compute the assignment matrix, the text and video projection layers from the global representation in Figure 2(c) are used to project multimodal features into a common space for feature similarity calculation. To ensure that the word-to-frame assignment contains more diversity instead of just saturated assignments to a single video frame, we add a constraint that requires label assignments to be equally distributed across various video frames representing diverse object/action concepts. This is achieved by restricting $\mathbf{Q_v}$ to a *transportation polytope* $\mathcal{Q}_v$:

$$\mathcal{Q} = \left\{ \mathbf{Q} \in \mathbb{R}_+^{U \times K} \mid \mathbf{Q}\mathbf{1}_K = \tfrac{1}{U}\mathbf{1}_U, \mathbf{Q}^\top \mathbf{1}_U = \tfrac{1}{K}\mathbf{1}_K \right\}, \quad (2)$$

which enforces the soft-assignment distribution $\mathbf{Q}$ to assign an equal marginal probability to each of the U frames instead of converging to a single frame. The vector $\mathbf{1}_U$ represents one vector with dimension $U \times 1$.

The next goal is to enforce this *transportation polytope* $\mathcal{Q}$. A solution for $\mathbf{Q}$ is now computed using the optimal transport Sinkhorn-Knopp algorithm [7, 11] as shown in Figure 2(b). The Sinkhorn-Knopp algorithm also normalizes the distribution of $\mathbf{P}$ as:

$$\mathbf{Q} = \mathrm{Diag}(\alpha) \exp\left(\tfrac{\mathbf{P}}{\varepsilon}\right) \mathrm{Diag}(\beta), \quad (3)$$

where α and β are scaling vectors that restrict $\mathbf{Q}$ to have a uniform distribution across region assignment. ε is a parameter that controls the smoothness of the mapping [7].

The T frames are then selected by the corresponding assignment $\mathbf{Q}$ from the frames with top T aggregated similarity sum over each word for further training. Note that the selection part $\mathbf{P}$ is from a trainable projection. While acquiring a better word-to-region projection during training, we hypothesize that the frame selection also benefits. The respective frame selection strategy is evaluated in Table 5.

8. Experimental setup

8.1. Baseline details

MIL-NCE [36], which utilizes S3D [53] and word2vec [37] to project two modalities into a common space, is chosen as the standard baseline for this task; CoMMA [46], the best-performing model for spatial representations in self-supervised learning (we denote CoMMA† to represent the model that uses weights shared by the author[3]); CLIP [38], an image-text model trained with transformer architecture, is further applied as the backbone and trained with [46] to construct CoMMA‡; GLIP [31] and RegionCLIP [62], state-of-the-art image-text grounding models that combine large-scale image caption pretraining and object detection fine-tuning, which we consider weakly supervised as the bounding box proposal network was trained on other human-annotated data. We further construct a strong baseline out of the best methods for temporal and spatial localization, MIL-NCE+RegionCLIP, where we use MIL-NCE for temporal localization and RegionCLIP for spatial grounding following the inference pipeline of Figure 3 without additional training.

8.2. Backbones and Training

We evaluate the proposed method on backbones, CLIP [38] and S3D-word2vec [36]. We described the detailed setup as well as the training in the following.

CLIP models. For both the visual and text backbone, we use the pretrained weights from CLIP [38] with transformer ViT-B/32 and fix the encoder. Both the visual and text encoder has a final embedding size of 512. We apply them to video segments with 12-28 seconds, processing 1 frame per second. An evaluation of how many frames to process (identical to the number of seconds) is shown in Table 9. We sampled the video with 5 fps. It shows the best results when we start with 80 possible frames U (as described in Section 3.2), from which T = 16 frames are selected for training. Ablation of the number of frames T used for training is shown in Table 10. We used a batch size of B = 64 video clips.

S3D-word2vec models. For the video backbone, we follow [46] and use S3D initialized by MIL-NCE on HowTo100M [36] at the rate of 5 frames per second and fix the video encoder. The global video clip features were max-pooled over time and projected into embeddings of dimension 512. We used the mean-pooled S3D spatio-temporal features to represent the global representation of the video following the S3D architecture [53]. For the text feature, we follow [35] using a GoogleNews pre-trained word2vec model [37] and max-pooling over words in a given sentence to acquire the text global feature. We follow [36] to use the max-pooled word embedding to represent the sentence

[3] We thank the authors for providing code and weights.

(global representation) since there is no [CLS] token. Also, the sentence feature is used for the query word selection instead of the [CLS] token. We use a batch size of $B = 96$ video clips.

Training. For the training of both backbone settings, we use an Adam optimizer [26] with a learning rate of $1e{-}4$. In the setting of fintining CLIP, we set a learning rate of $1e{-}7$ for the CLIP backbone. The model is trained for 10 epochs on 4 V100 GPUs, which takes about two days.

8.3. Inference

Inference for the proposed model and CoMMA. For inference in the case of temporal grounding, as shown in Figure 3(a), we first normalize the global feature for video and text. We used a (temporal) threshold $\theta = 0.5$ to separate detections from the background. In spatial grounding, we acquire an attention heatmap using the attention rollout [1] described in Section 3.5. We set a spatial threshold $\tau = 0.01$ to create the mask, as shown in Figure 3(b). The choice of this spatial threshold is evaluated in Table 12.

GLIP, RegionCLIP baseline inference. In spatial grounding, we are given a text query and need to localize it in the frame. GLIP and RegionCLIP predict multiple bounding boxes corresponding to the text query. We select the predicted bounding box with the highest confidence score as the prediction result. We use the center point of the predicted bounding box for the pointing game evaluation as the model prediction. For *mAP* evaluation, we use the predicted bounding box to compute IoU with the ground truth bounding box. In spatio-temporal grounding, we input all possible action description labels as candidates similar to Figure 3(a). We pick the class with the highest confidence score as the predicted label. If the model made no prediction, we would predict it as "background". The spatial inference is the same as the spatial grounding setting.

TubeDETR, STCAT baseline inference. TubeDETR and STCAT are spatio-temporal grounding models trained to predict a single spatio-temporal tube per video. In both cases, TubeDETR and STCAT, we use models trained on the Vid-STG dataset with 448x448 resolution and evaluate them for the task of spatial grounding. Since this dataset contains mostly short videos (<30sec), we observed that both methods will also only predict a trajectory tube in this temporal range (<30sec), no matter how long the input video is. To allow us to apply them to longer videos (>30sec), we split the longer videos based on sliding windows of 5-sec for better performance.

MIL-NCE, CLIP baseline inference. Both models are trained based on global representations for both input modalities, videos/images and text. We can, therefore, directly compute a sentence-to-video-frame similarity to perform the temporal grounding for Figure 3(a), following the same process as the proposed method for temporal grounding. For spatial grounding, we compute sentence-to-region feature similarity. Both visual backbones produce a 7x7 grid feature. We normalize the sentence and region features, then select a spatial threshold $\tau = 0.5$ to create the mask for the *mAP* evaluation.

8.4. Evaluation metrics

(i) **Spatio-temporal grounding in untrimmed video** is evaluated on our annotated GroundingYoutube dataset. We combined the spatial and temporal grounding evaluation as before [2, 28] to form the spatio-temporal evaluation. The entire video and the respective pool of action instructions were provided. The model needs to localize each action step in temporal (start-time/end-time) and spatial (location in the video) as described in Figure 3. We evaluate in two metrics: **IoU+Pointing game** combines the evaluation setting from the spatial grounding [2] and temporal grounding [28] metrics. For each video frame, the prediction is correct when the model predicts the correct action for the frame. Also, given the predicted action as a query, the maximum point of the heatmap aims to lie within the desired bounding box. We then compute the Intersection over Union (IoU) over all the predictions with the GT to acquire the final score. We also compute **video mAP** following previous evaluation [15], where we set IoU threshold between GT and predicted spatio-temporal tubes. A prediction is correct when it surpasses the IoU threshold. We then compute the mAP over all classes. We form a 3D prediction mask following Figure 3 and compute IoU between our 3D heatmap and 3D tube.

(ii) **Spatial grounding** is given a text query description to localize the corresponding region in the trimmed video. We use GroundingYoutube, Youcook-Interaction, V-HICO, and Daly for evaluation. This task is evaluated using the **pointing game accuracy**. Given the query text and video, we compute the attention heatmap on the video as described in Figure 3(b). If the highest attention similarity score lies in the ground truth bounding box, the result counts as a "hit" and counts as "miss" otherwise. The final accuracy is calculated as a ratio between hits to the total number of predictions $\frac{\#\text{ hits}}{\#\text{ hits}+\#\text{ misses}}$. We report the mean average precision **(mAP)** following the settings from V-HICO [32]. Given a human-object category as the text query, we aim to localize the spatial location in the video frame. The predicted location is correct if their Intersection over-Union (IoU) with ground truth bounding boxes is larger than 0.3. Since we do not use any bounding box proposal tools or supervision, we create an attention heatmap as described in Figure 3(b) to create a mask for IoU computation. We follow [32] and compute the mAP over all verb-object classes.

(iii) **Temporal grounding** provides videos with the respective actions and their ordering, including the background. The goal is to find the correct frame-wise segmentation of the video. We follow the inference procedure in [28]

to compute the alignment given our similarity input matrix. The task is evaluated by intersection over detection (IoD), defined as $\frac{G \cap D}{D}$ the ratio between the intersection of ground-truth action G and prediction D to prediction D, and the Jaccard index, which is an (IoU) given as $\frac{G \cap D}{G \cup D}$.

9. Additional Experiments

9.1. Runtime analysis

We analyze the computational costs of sampling and loss. We sample 16-second videos at a frame rate of 5 FPS (80 frames in total). We report the execution time for a single batch (batch size = 64) averaged over 100 batches. For the *frame sampling strategy*: (1) Random select 8 frames: 1.48s. (2) Optimal transport based selection of 8 frames out of 64: 1.54s. (3) Entire 64 frames: 1.74s. The execution time of our method is close to traditional random sampling while capturing diverse visual concepts, which improves the training process. For the *global and local* components: (1) Global loss only: 1.1s. (2) Local loss only: 1.52s. (3) Both losses: 1.54s. Computation of the local loss is more time-consuming than the global loss due to its requirement for features with finer granularity.

9.2. Single-action spatio-temporal grounding.

Current spatio-temporal detection and grounding datasets [15, 22] usually aim to discriminate a single given action class from the background class in a short clip. This differs from our setup of spatio-temporal grounding in untrimmed videos, which usually comprises a set of phrases that need to be detected in a 3-5 min long video. To allow an evaluation of spatio-temporal grounding approaches based on single phrase grounding, we construct a clip-level evaluation where the clip varies from 9 sec to 60 sec. Given an action step, we append the video segments before and after the steps with the same time length of the action step to form the final video clip. This results in 2,895 clips for the spatio-temporal clip grounding evaluation. For each clip, the temporal action intervals occupy 33% of corresponding videos, which demonstrates the difficulty of the setting. In this setting, instead of selecting the possible action step from a pool, the ground truth action step was given as the text query for spatio-temporal grounding. This allows us to directly compare with supervised spatio-temporal grounding methods [23, 54] as described in Section 5.4. As shown in Table 7, we observe that the baseline GLIP models achieve a much better performance compared to Table 2. This is due to the fact that this setting does not require the model to select the text query from the pool, which the GLIP model was not trained to do. Moreover, we find that weakly supervised methods, GLIP and RegionCLIP, show only limited ability to differentiate the queried action from the background, which leads the model to ground the text query in most of the frames. However, both demonstrate powerful localization ability in foreground action segments, which results in a decent performance. The fully-supervised trained models (TubeDETR, STCAT) achieved a balance in localizing temporally and spatially, resulting in the best performance on this task.

9.3. Ablation and decision choices

We performed additional ablation studies using the CLIP backbone without finetuning.

Attention architecture. We tested different architectures by stacking the self-attention or cross-attention block in the model to calculate contextualized local representations, as shown in Figure 2(d). As shown in Table 8, we found that the standard multimodal transformer architecture (self+cross) to have the worst performance. Using two cross-attention blocks was beneficial in incorporating more cross-modal interaction between local features. Finally, including a self-attention layer slightly improves the final representations by encoding better single-modality representations.

Frames used for selection. As shown in Table 9, we perform an ablation study on the number of candidates frames U used for training. We found that selecting 80 frames (16 seconds) achieves the best performance, comprising the useful video information in training while not including too many irrelevant concepts that diverge from the action/object in the ASR sentence.

Number of frames for training. We further evaluated the impact of different numbers of frames T used for training. As shown in Table 10, selecting fewer frames for training significantly causes the performance to drop. We hypothesize that the model not only fail to capture the temporal dynamics with fewer frames but also loses some frames with groundable objects in the sentence while training. We also hypothesize that with a too large number of frames, more irrelevant frames might be selected during training, which decreases the performance.

Effect of audio in training and testing. Unlike text which describes a discrete concept as a target to ground, audio serves as a continuous representation that is highly relevant to the temporal information. For example, we can determine an action started when we hear a "cracking" sound. In Table 11, we tested our model using the additional audio modality. For the audio branch, we compute log-mel spectrograms and use a DAVEnet model [19] initialized by MCN on HowTo100M [8] to extract audio features. We extend the global and local loss pairs from VT to VT, VA, and AT following [42]. We found when training and testing with audio, the spatio-temporal result increases the temporal performance while the spatial-only result remains the same. This validates our assumption that audio contributes more to temporal understanding. When we trained on audio

Method	Backbone	DataSet	Supervision	Modality	GroundingYoutube						
					IoU+Point	mAP 0.1	0.2	0.3	0.4	0.5	0.1:0.5
CoMMA* [46]	S3D-word2vec	HT250K	Self	VT	1.10	7.46	5.84	4.20	2.65	1.53	4.93
MIL-NCE [36]	S3D-word2vec	HT100M	Self	VT	12.41	45.91	32.33	15.35	3.70	2.56	19.54
Ours	S3D-word2vec	HT200K	Self	VT	19.46	51.95	40.31	26.81	16.27	7.81	28.63
CoMMA† [46]	CLIP	HT200M	Self	VT	2.64	8.94	6.89	5.47	4.18	2.67	5.63
CLIP [38]	CLIP	HT200K	Self	IT	11.34	43.28	30.64	11.20	3.10	1.94	18.03
RegionCLIP [62]	ResNet-101	CC3M	Weak	IT	17.42	51.86	40.23	26.10	15.23	7.29	28.14
GLIP [31]	Swin-L	Cap24M	Weak	IT	18.15	52.61	41.83	26.93	17.23	8.46	29.41
Ours	CLIP	HT200K	Self	VT	20.81	53.24	42.96	29.17	20.36	11.84	31.51
TubeDETR [54]	MDETR	Vid-STG	Full	VT	26.43	63.47	50.95	38.23	28.31	19.34	40.06
STCAT [23]	ResNet-101	Vid-STG	Full	VT	27.84	64.96	52.13	40.61	30.49	20.55	41.75

Table 7. **Single-action spatio-temporal grounding in short videos.** We compare spatio-temporal grounding approaches based on single phrase grounding. To this end, we construct a clip-level evaluation based on the action segments of GroundingYouTube, where each action segment varies from 9 sec to 60 sec. We append video segments before and after the annotated action with the same time length of the action step to form the final video clip. This allows us to directly compare with supervised spatio-temporal grounding methods [23, 54].

Attention Architecture	GroundingYT Spatio-temporal	MiningYT Temporal	YouCook-Inter. Spatial
Self+Cross	15.4	18.7	54.1
Cross+Self	15.9	18.9	54.5
Cross+Cross	16.5	19.3	56.2
Cross+Self+Cross	17.1	19.9	57.1

Table 8. **Ablation on different attention architecture**

# of frames	60	80	100	120	140
GYT (Spatio-temporal)	16.4	17.1	17.0	16.8	16.1
YC-Inter (Spatial)	56.3	57.1	56.8	56.7	55.9

Table 9. **Ablation of # of frames used for selection**

Frame length	1	4	8	16	24
GYT (Spatio-temporal)	5.2	9.5	16.1	17.1	16.5
YC-Inter (Spatial)	31.1	48.2	55.5	57.1	56.1

Table 10. **Effect of # video frames used for training**

Train/test supervision	VT/VT	VAT/VT	VAT/VAT
GYT (Spatio-temporal)	16.2	16.8	17.0
YC-Inter (Spatial)	53.9	53.6	53.8

Table 11. **Effect of audio supervision in train and test**

and tested without audio, the performance increased over the VT model, showing that the audio serves as useful supervision for better video/text representations.

Threshold for attention mask. As shown in Figure 3(b), we apply a threshold to create a mask from the result of attention rollout. Note that this threshold τ is not a hyperparameter that affects the training or the model but simply serves as a means to an end to compute the *mAP* scores.

Treshold	Backbone	0.1	0.05	0.01	0.005	0.001
CoMMA*	S3D-word2vec	0.76	0.90	0.93	0.91	0.86
Ours	S3D-word2vec	15.35	15.88	16.22	16.34	16.12
CoMMA†	CLIP	0.88	0.92	0.99	0.94	0.91
Ours	CLIP	15.93	16.33	17.10	17.05	16.24

Table 12. **Threshold for attention score on GroundingYoutube *mAP*@0.4**

	GroundingYT Spatio-temp	MiningYT Temporal	YC-Inter. Spatial
Mean-pooling	18.39	19.24	57.84
$[CLS]$	19.45	20.33	58.35

Table 13. **Global representation ablation with CLIP.** $[CLS]$ showed superior performance due to self-attention architecture.

	GroundingYT Spatio-temp	MiningYT Temporal	YC-Inter. Spatial
only Local loss	5.62	4.97	50.54
only Global loss	7.53	18.67	31.51
w/ Both loss	16.22	19.18	53.98

Table 14. **Loss ablations with MIL-NCE (S3D).** Results show a similar trend as the CLIP backbone used in the main paper.

We did not systematically optimize this threshold, but instead, Test different thresholds for attention scores for all relevant models (COMMA, ours) using the spatio-temporal grounding *mAP* IoU@0.4 on our GroundingYoutube dataset as shown in Table 12. We find 0.01 to be a reasonable threshold among all models, performing best on COMMA and giving at least the second best results for the proposed model.

Mean pooling for global features We also tried mean pooling over all tokens for CLIP to replace [CLS] for the global feature. As shown in the tab. 13, [CLS] outperforms mean pooling in our 3 datasets. We attribute this to the fact that [CLS] was calculated by self-attention, which will automatically select important tokens, whereas mean pooling treats all tokens with the same importance.

Loss ablation with MIL-NCE(S3D) We tested the local vs.

global loss on top of S3D (initialized by MIL-NCE, see Tab. 14) showing similar behavior compared to CLIP.(shown in Table 6 in the main paper)

10. GroundingYoutube Annotation

The data annotation was divided into three phases: During *Phase I* (Sec. 10.1, a graphical user interface (UI) and the task description were developed. In *Phase II*, the dataset was given to the annotators to generate the key points (Sec. 10.1). In *Phase III*, a manual quality control step was performed (Sec. 10.2).

10.1. Development of the graphical user interface and task description

The annotation of a large amount of data is often one of the most expensive aspects of a machine learning pipeline design, which is why the annotation time per datum should be kept as short as possible. There are two points that can be optimized, (1) the training or the task "message" for the annotators and (2) the graphical user interface by minimizing interaction times.

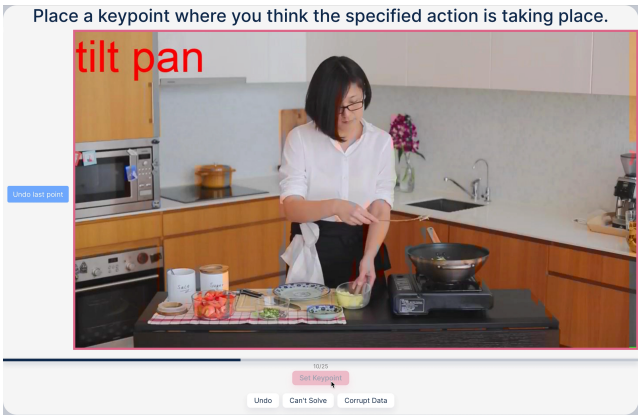


Figure 6. **A screenshot of our simplified annotation interface.** On the top, the annotation task is described in simple and short words to save reading time. To make interacting with the UI as intuitive as possible, actions are limited to simple button clicks and setting the key point by clicking on the image.

While tasks are usually formulated in such a way that no ambiguities arise, i.e. all possible edge cases are somehow covered, and simple words are used, in this case, we made a conscious decision to choose questions as short as possible, and that would give the annotator room for interpretation. We did this because it was hard to predict where people would actually locate actions in images. We also created a 1 min 30 sec long user training video where we demonstrate the task using exemplary keypoint annotations and explain how to use the UI.

Our annotation UI was designed with a special focus to keep it as intuitive as possible and reducing the interaction time. Our UI only provided five functionalities (set/unset a keypoint, undo the last image, image can't be solved, and image is corrupt) which were clearly described in text buttons (see Figure 6). Further, to reduce the cognitive load of our workers, images were presented in the form of work packages, each containing 25 images. Hence, we could ensure that completing a task would take no longer than 6 minutes.

The annotation of all $26,987$ images was performed with five distinct repeats per image, resulting in $134,935$ labels in total. All labels were generated by 13 professional annotators in total, which took them $5s$ in average per image. However, it should be noted that the number of images where an annotator placed a keypoint differs along all the workers (see Figure 7) and that the vast majority of all images have been answered by five annotators only. Examples are shown in Figure 8.

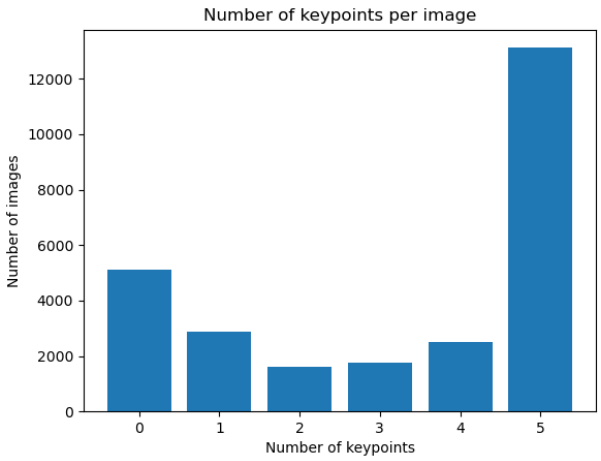


Figure 7. **Number of keypoints per image**. It can be seen that 48% of the data has all 5 key points and 19% has not a single annotation

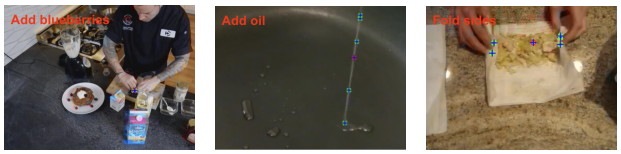
During the annotation, professional annotators were given a short instruction video at the beginning and then asked to click on the center of the given action without additional instructions. They were further free to choose "can't answer" if they could not locate the action, e.g., at the beginning and end of the clip. Thus, the number of available key points per image differs, and we choose majority voting to determine whether an action is present, resulting in new, refined temporal boundaries compared to the original annotation.

We found that the point-wise annotation resulted in roughly three distinct patterns, which depend on the captured scenario, as shown in Figure 9. In the case of half portrait or even wider shots in Figure 9a, annotations are highly locally centered. We further found that in some cases, the

(a) Can't solve (b) Single point

(c) Four points (d) Five points

Figure 8. **Sample annotations**. The purple point represents the center point of the annotations in the frame. 48% of the data has all 5 key points, and 19% has not had a single annotation.



(a) Wider shot (b) Flow action (c) Split to two

Figure 9. **Example of keypoint annotations under different conditions.**

point annotation can also represent the flow of the action, e.g., pouring oil in Figure 9b, or even split into two separate clusters in Figure 9c.

10.2. Quality control

Since the label quality of the datasets used is a critical factor in the performance of machine learning models, we verified the correctness of a subset of our images using an experienced annotation specialist for $1,026$ randomly selected frames. To evaluate the data quality, we evaluate the agreement between the annotation specialist and the annotations provided by the annotators. To this end, we considered an annotation as a false positive if three annotators or more have set a key point, although no action can be seen in the image, and as a false negative if three annotators or more have not set a key point, even though an action can be seen in the image. The entire sample was assessed using these criteria, with the specialist disagreeing with the annotators in only a total of $1.1\% \pm 3\%$ (FP: $0.7\% \pm 3\%$, FN: $0.4\% \pm 3\%$). We also found that annotations significantly diverted in terms of spread. Namely, wider shots tend to be highly centered, whereas zooming in together with the usage of larger objects such as a pan or a spatula results in more widespread key points. We also analyzed how often those cases occur and found that 14.0% of the selected frames show a widespread pattern.

Sample size calculation To this end, we first needed a representative subset of N_S images of our data. We calculated the required sample size based on the following two formulas:

$$N_0 = \frac{z^2}{\epsilon^2} \cdot p \cdot (1-p) \tag{4}$$

where α is the confidence interval, p the expected probability of the appearance of a quality aspect (e.g., widespread answers), $epsilon$ is the accepted error margin, and $Q(\alpha)$ is the percent point function of a normal distribution and $z = Q(1 - \frac{\alpha}{2})$.

As N_0 would be the required sample size for an infinitely large population, we applied the finite population factor that results from sampling without replacement from a finite population.

$$N_S = \frac{(N_0 \cdot N)}{N_0 + (N-1)} \tag{5}$$

where N is the total number of images.

We set $\alpha = 95\%$, $epsilon = 3\%$, and our sample size of $N = 26,987$. As the probability of the quality aspect is unknown, we set $p = 50\%$, which resulted in $1,026$ being checked for quality control.

Distribution Type	*mAP*@0.4
Widespread actions	18.34
Saturated actions	15.96
Total	17.10

Table 15. **Performance on the annotation distribution types of widespread v.s. saturated.**

10.3. Dataset usage for evaluation

How to derive bounding box? We derive bounding boxes by adding a constant distance ($0.15\times$frame_width in width, $0.15\times$frame_height in height) to the boundaries of a union of all annotated points (as shown in Figure 4). Since point annotations may be scattered in the image or, conversely, gathered around a point, the output bounding boxes will vary in size over time and for different actions. We manually check the auto-generated bounding boxes and adjust the bounding box when needed.

Performance on widespread and saturated action. We evaluate the performance of different action distributions using the spatio-temporal grounding *mAP* IoU@0.4 setting. We define widespread actions to have an area larger than a certain threshold A. Here, we set $A = 60,000$ pixels. As shown in Table 15, the performance of the widespread actions was higher since it had a higher tolerance of spatial localization error.