APPENDIX

Here we provide our parameter study results and the extended version of figures that we shortened in the paper. In Fig. A1, we present the results of the parameter study, where we vary the size of the spatiotemporal neighborhood (offsets o_s, o_t) and measure the resulting query accuracy. Despite significant changes to the parameter values, the model's performance remains consistent across several different queries, especially for the larger queries.

In Fig. A2, we show a larger version of Fig. 3, including more of the best-matching timesteps. As we see in Fig. A2a and Fig. A2b, the model successfully finds many examples of deposition and splashing, showing homogeneous results overall. In Fig. A2c, we show a different non-filtered version of the fluid column query. When searching for a rare event with a single example patch, we get noisier results but still find other members displaying the same behavior (ID 71 and 62).

Fig. A3 is an extended Fig. 5 from the paper, showing more matches for both our model and the SIFT-based method on the droplet ensemble. We observe that the results are consistent for these matches, i.e., both methods perform well on the crown query, but our model produces more robust results on the splashing query, where SIFT is unable to find good keypoint locations.

Fig. A4 displays our prototype system on the "droplet splash" ensemble. Here we search for examples of splashing, which we express with a small query that has positive splashing and negative crown examples. Similarly to the results in Fig. A2, we see that our search method finds other cases of splashing in the ensemble. Like in the cylinder example from the paper (Fig. 2), the model also roughly determines the point of transition from a crown to a spray, which happens at different temporal locations in different ensemble members. This suggests our matching technique is able to locate events temporally.

Figures A6 and A7 are straightforward extensions of paper figures 8 and 9 respectively. The difference here is that we include more query sizes ("2+0-" and "2+2-") and show the performance of additional methods (Hist-L1, SSIM) and encoder models (VGG-tune-a, VGG-tune-l). The additional query sizes show a gradual change in the model's performance as we change the query size: larger queries generally lead to better accuracy and reduced variance both wrt. query (A6) as well as the training process (A7). Hist-L1 is similar to Hist-EMD, but we use the L1 distance between the histograms, and SSIM is the negative of the Structural Similarity Index Measure (we require a dissimilarity measure for our ranking score). We see that these methods perform similarly to other generic baselines and are outperformed by our model. VGG-tune-a and VGG-tune-l are the pre-trained VGG models that were fine-tuned by using them as the encoder in our siamese architecture, trained on the pretext task as usual. For VGG-tune-a, we fine-tuned all the layers of the model (up to "fc1", which we use as the encoder output for VGG), while for VGG-tune-l, we only tuned the last layer. Since we are using our self-supervised setup to fine-tune the VGG model, this is essentially just a different larger architecture of the encoder that was pre-trained on a generic image dataset. As we can see in Fig. A6, fine-tuning on the pretext task improves the performance of the VGG model. However, our smaller encoder still performs better overall, especially on the larger queries. Conceptually, VGG is very similar to our encoder (a rather standard convolutional model), so it performs similarly but likely suffers a little due to being pre-trained on a significantly different image dataset and is just too large for the task at hand.

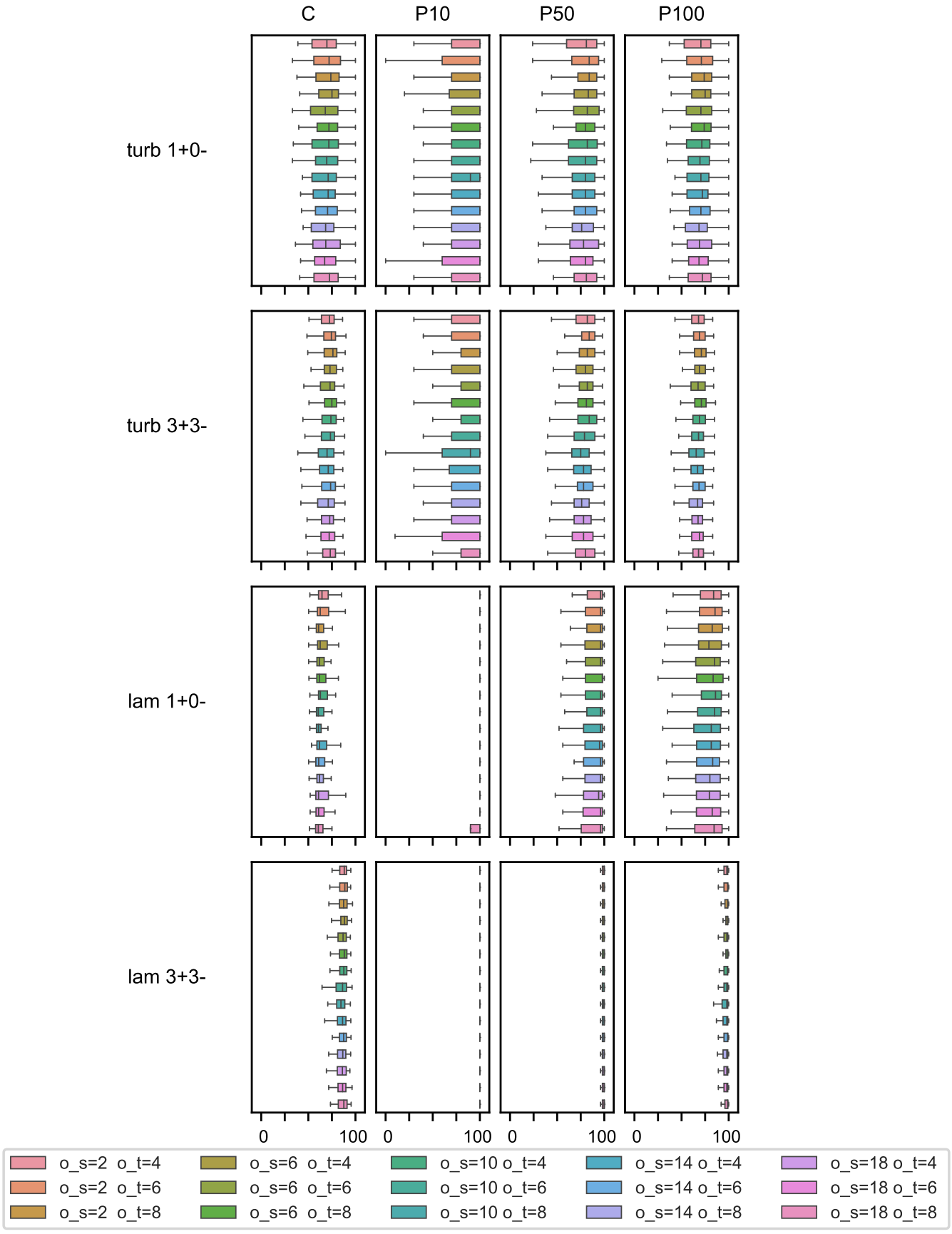
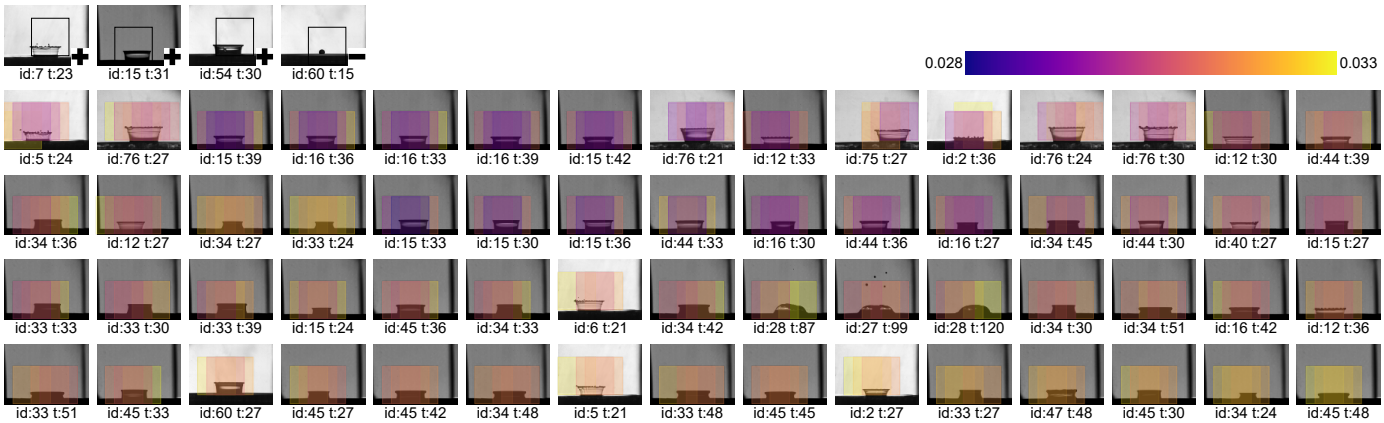
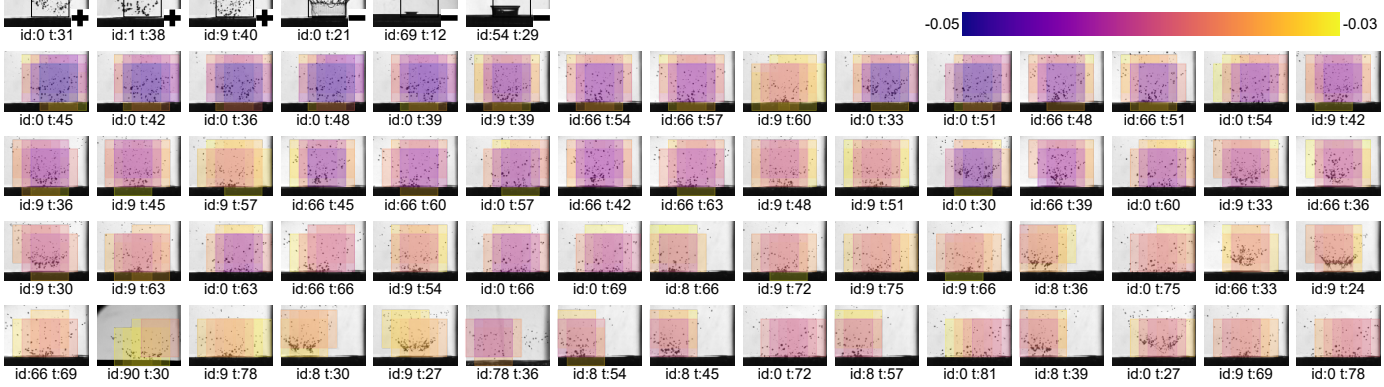


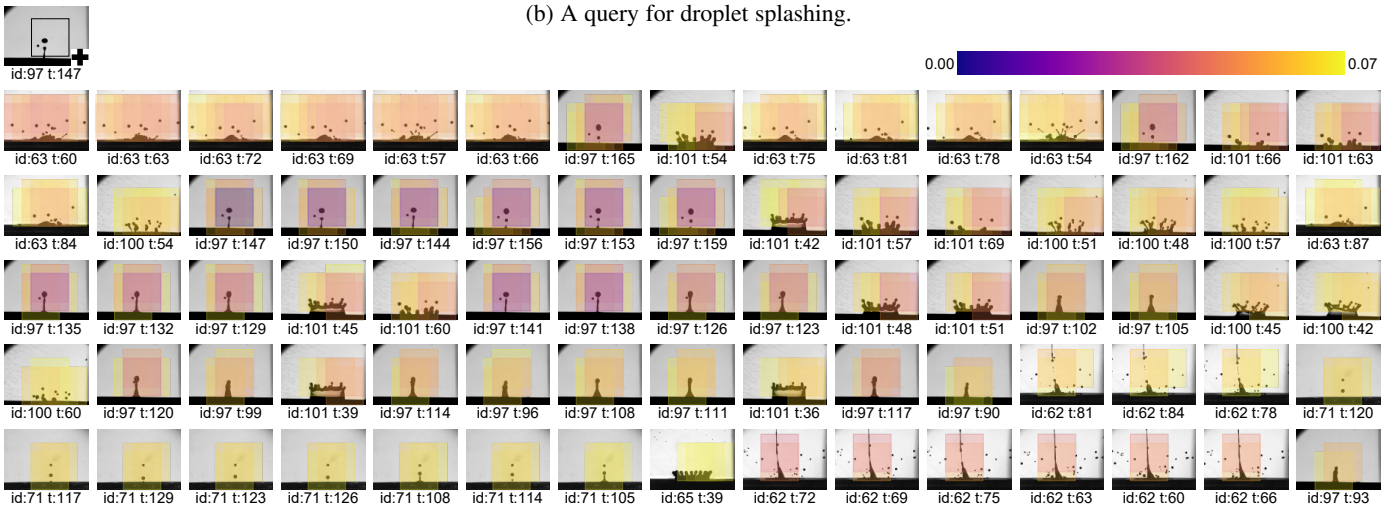
Fig. A1: Results of a parameter study on the cylinder ensemble. The experiment setup is similar to Fig. 8 with metrics excluding the query members to better assess the generalization performance. Despite the changes in both the spatial and temporal offsets used during the model training, all models produce similar results. This indicates that our method is not overly sensitive to the parameters, which can be chosen with some minor knowledge of the dataset size.



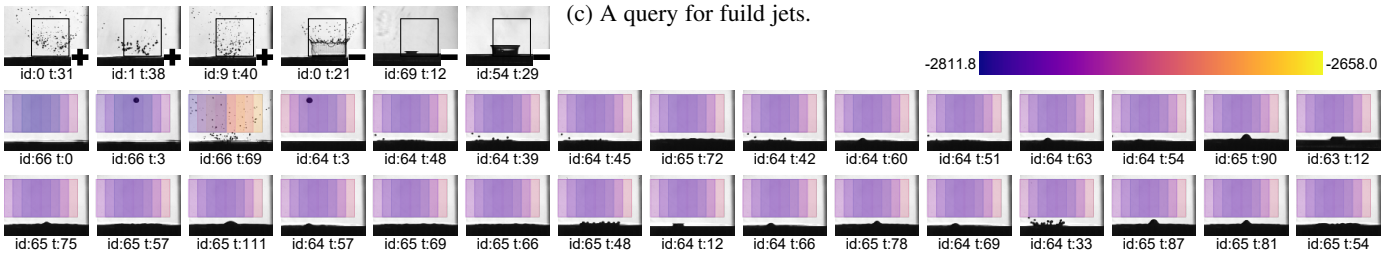
(a) A query for crowns.



(b) A query for droplet splashing.

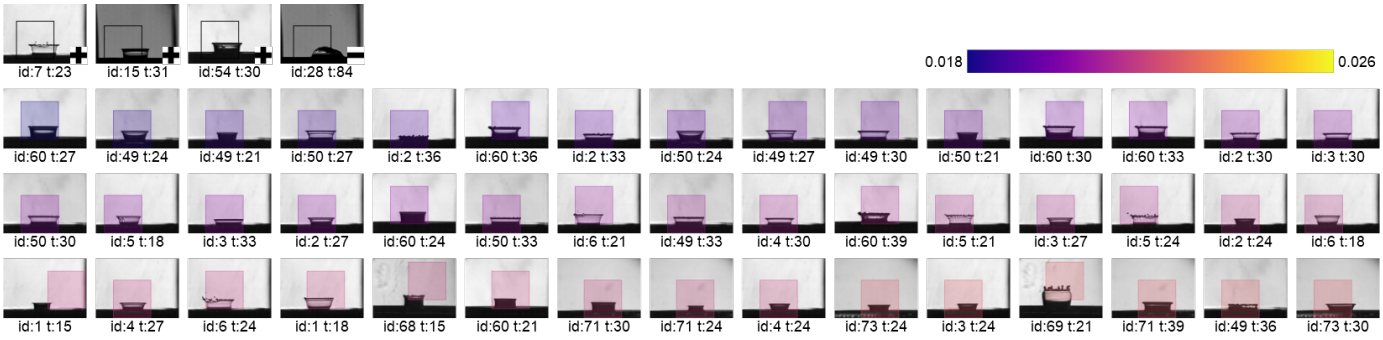


(c) A query for fluid jets.

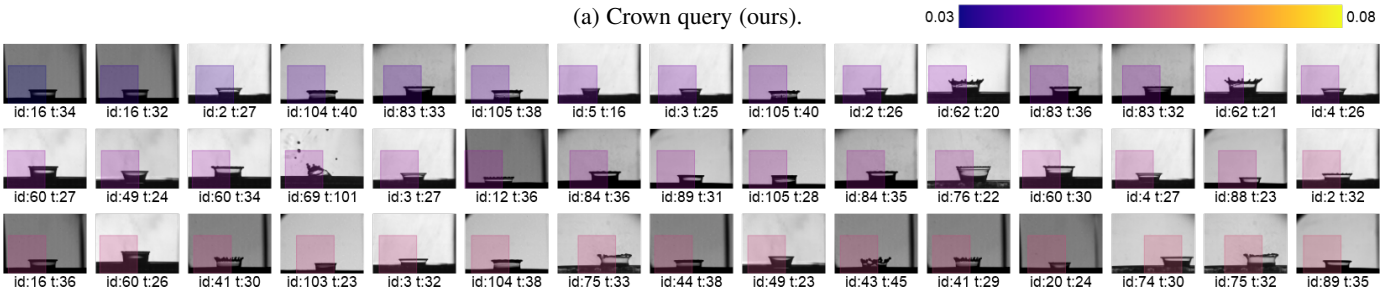


(d) MSE results for the query from **b**.

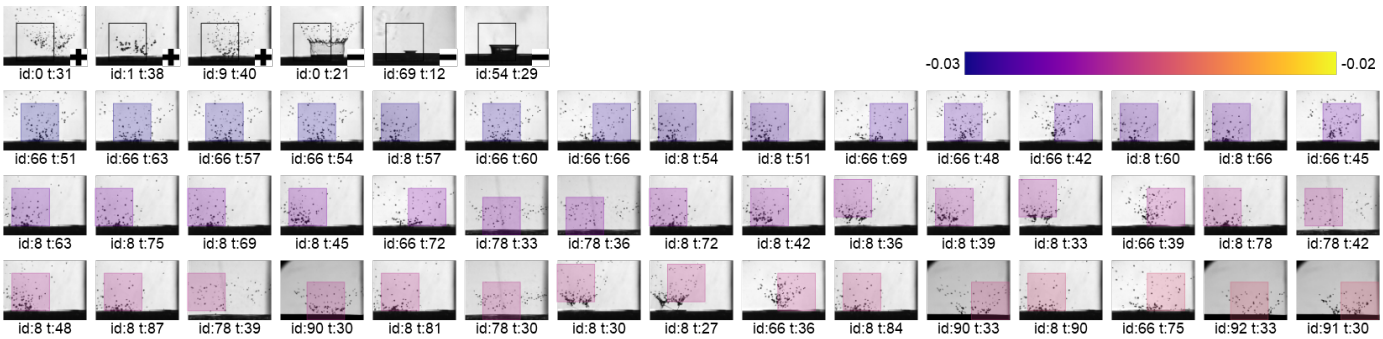
Fig. A2: An extended version of Fig. 3. Query results on the droplet splash dataset. In the top left we render the query patches (their first timestep). Positive examples are marked with **+** and negative with **-**. We take 500 best matching patches and render the timestep they start on, sorted by the number of matches in that timestep (i.e. frames with most best matches). The member ID and the timestep index are printed below each frame. We color the matching patches based on their score, where blue means better matches. As we see in **a, b**, our method finds many diverse examples of the queried behavior. Even with a single patch of rare behavior (**c**) we can find its other instances. Compared to the MSE baseline in **d**, our method returns much more useful results.



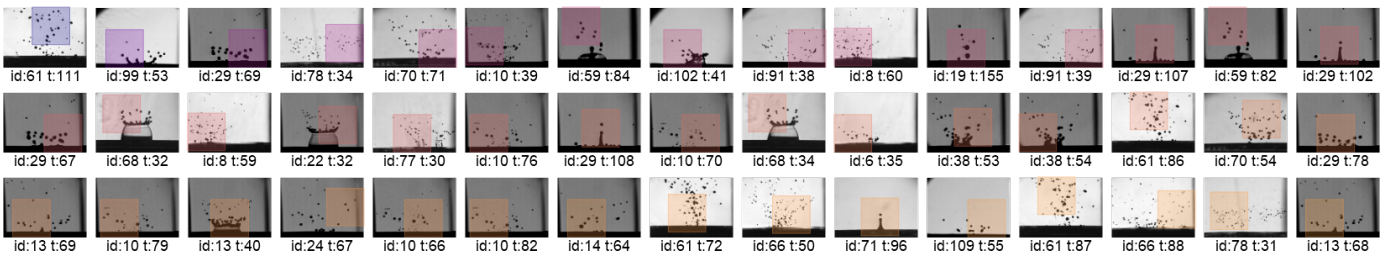
(a) Crown query (ours).



(b) Crown query (SIFT).



(c) Splashing query (ours).



(d) Splashing query (SIFT).

Fig. A3: An extended version of Fig. 5. Comparison of our similarity metric to SIFT on the droplet ensemble. **a, b:** we see that both methods produce good results on the fluid crown query, as it is very suitable for the SIFT descriptors. **c, d:** However, on the splashing query our method returns more robuts results, as SIFT is struggling to match small disperse droplets.

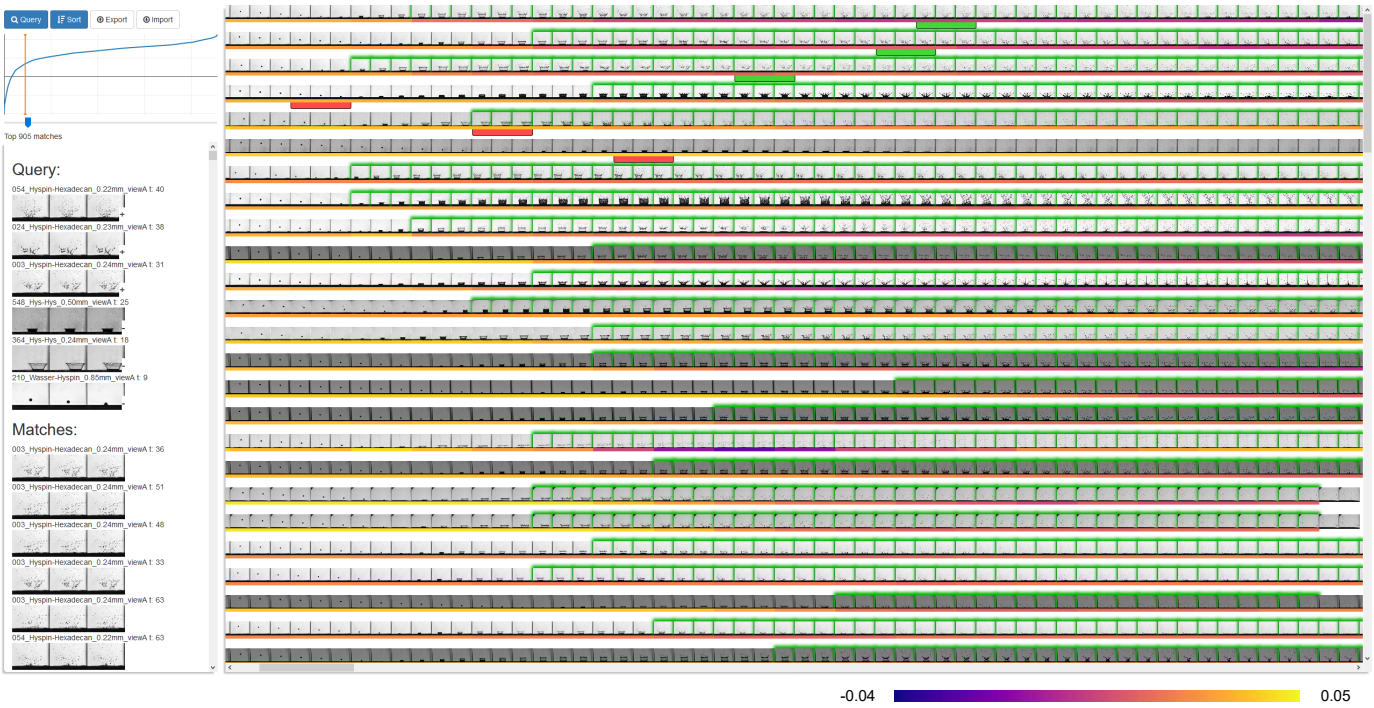


Fig. A4: An example query for splashing in the droplet splash ensemble. We provided three positive examples of spraying and three negative examples containing crowns without spray. The system successfully finds many other examples of droplet sprays in the ensemble. The model generalizes well beyond the query examples, matching instances with different spray shapes and scene illumination. We not only find members with splashing but see when the splashing transition happens temporally.

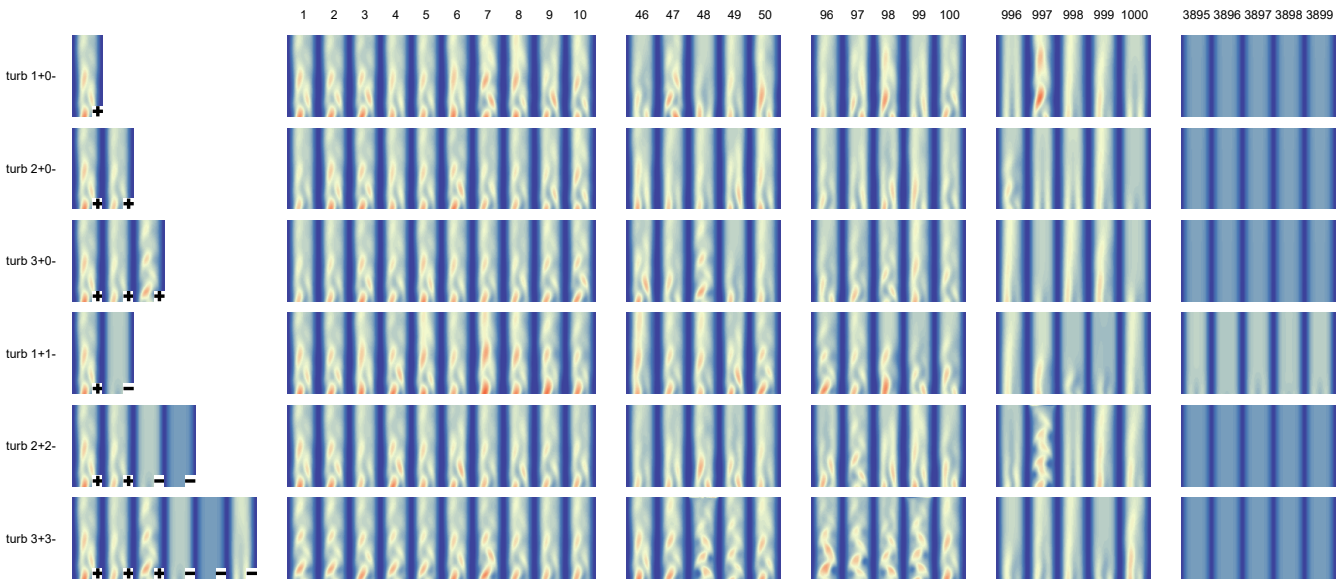
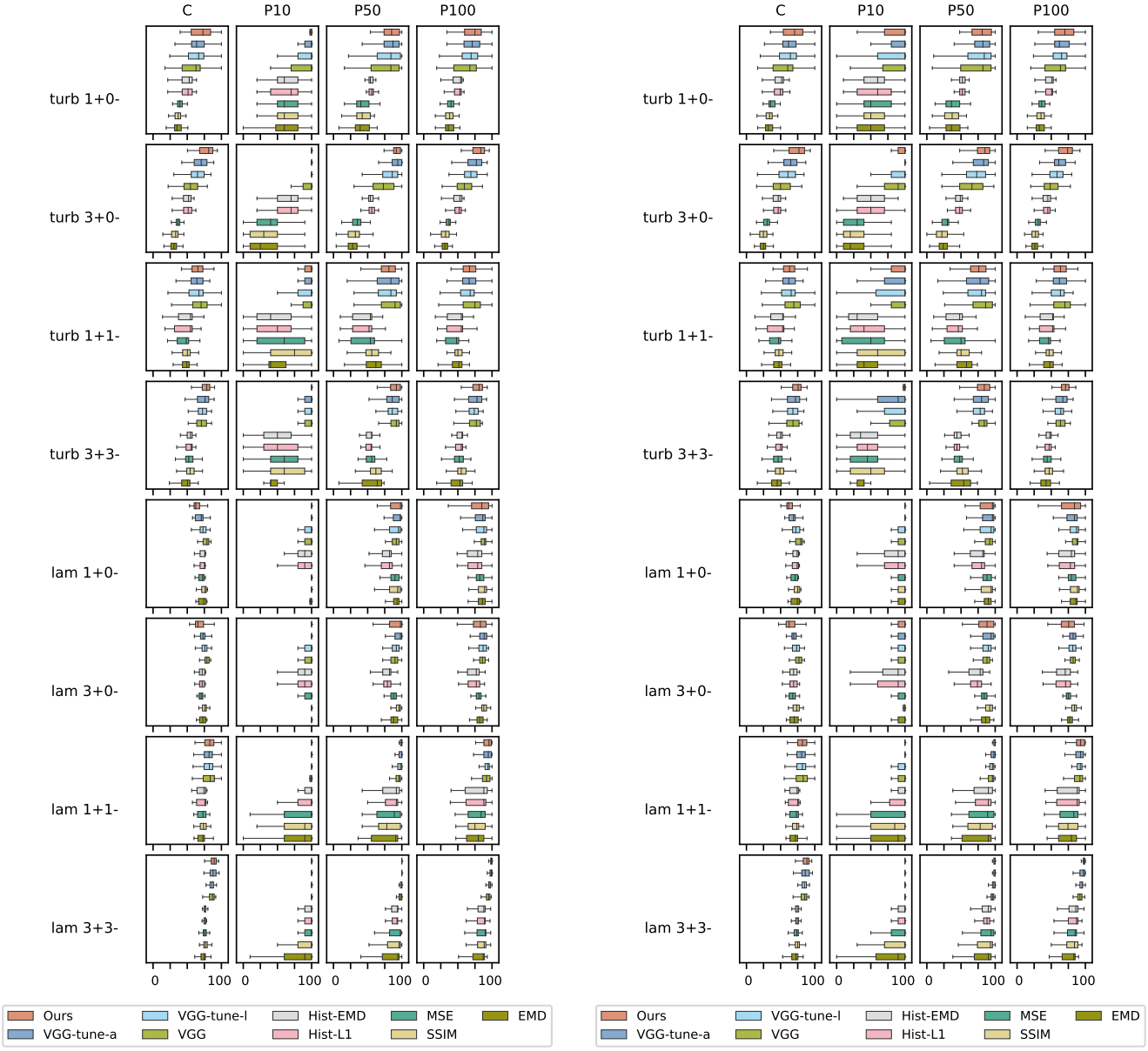


Fig. A5: An extended version of Fig. 7. Rendered matches to the turbulent queries from Tab. 2. Each row corresponds to a query. On the left, we show the patches in the query, and on the right, we show the matching patches from the ensemble. We only render the first frame of each patch. We show the top 10 matches as well as some examples of lower matches. The icon on the query patches indicates whether a patch is a positive (**+**) or a negative (**-**) example.



(a) Including all results.

(b) Excluding query members.

Fig. A6: An extended version of Fig. 8. We are measuring the distribution of search quality for “cylinder” wrt. random queries. Each row represents a query type, e.g., ’turb 3+1-’ means queries containing three random turbulent patches and one random non-turbulent patch. In the columns, we compute the same quality metrics as in Tab. 2, with all metrics ranging from 0 to 100. **a**: Our model shows better results than the baselines, especially on larger queries. We also see that the variance is reduced when more examples are used in the query. **b**: We evaluate the model’s generalization by excluding patches from members mentioned in the query. Performance is slightly worse (as expected when removing the best matches), but the model is still successful. This suggests that the model generalizes beyond the pretext task and finds other instances of behavior.

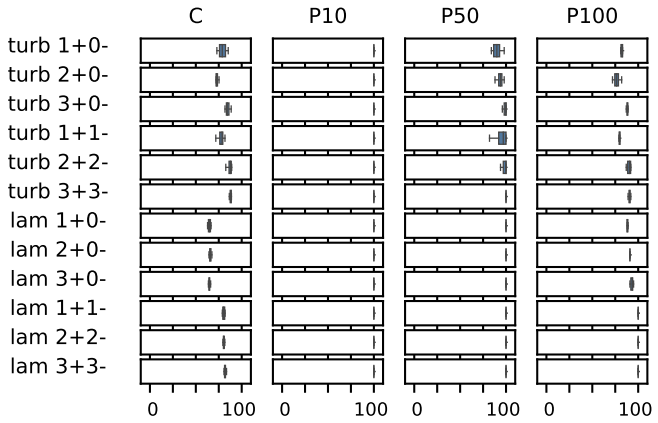


Fig. A7: An extended version of Fig. 9. The variance of search quality wrt. training process. Here we have trained our model ten times and performed the search with each one. We used the same manually constructed queries as in Tab. 2. While some variance is present, it is not significant and decreases with increasing query size.