Supplementary Material: Modeling Video as Stochastic Processes for Fine-Grained Video Representation Learning

Heng Zhang[1,2*] Daqing Liu[3*] Qi Zheng[4] Bing Su[1,2†]
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3] JD Explore Academy, JD.com [4] The University of Sydney
zhangheng@ruc.edu.cn, {liudq.ustc, subingats}@gmail.com, qi.zheng@sydney.edu.au

In this supplementary material, we first provide the preliminaries in Appendix A. Then, we give more quantitative results in Appendix B. Next, we present more qualitative results on the video alignment and fine-grained frame retrieval in Appendix C. Last, we discuss the hyperparameters setting in the practice and elaborate more details of the proposed method with an algorithm in Appendix D.

A. Preliminaries

Brownian motion. Brownian motion is also called the Wiener process, and the stochastic process $\{X(t) : t \geq 0\}$ is called Brownian motion. The standard Brownian motion satisfies the following conditions:

$$\begin{cases} X(0) = 0 \\ \{X(t) : t \geq 0\} \text{ is independent increment} \\ X(t) - X(s) \sim N(0, \sigma^2(t-s)) \end{cases} \tag{1}$$

where $\sigma = 1$ in the standard Brownian motion. Although the Brownian motion in VSP is not a standard one where the mean value is variable and $\sigma = \sqrt{\alpha(T-t)}$, it can be converted to standard Brownian motion by standardized transformation. The standard Brownian motion $\{B(t) : t \geq 0\}$ has the following important properties:

$$\begin{cases} \{B(t+\tau) - B(\tau) : t \geq 0\}, \ \ \tau > 0 \\ \{\frac{1}{c}B(c^2 t) : t \geq 0\}, \ \ c \neq 0 \end{cases} \tag{2}$$

The first equation is the Markov property while the second one is the self-similarity of standard Brownian motion.

Brownian bridge. Given a standard Brownian motion $\{B_t : t \geq 0\}$, let $X_t = B_t - tB_1$, the probability distribution is $X_t \sim N(0, t(1-t))$ then $\{X_t : 0 \leq t \leq 1\}$ is the Brownian Bridge process, which is a conditional stochastic process $\{B_t : 0 \leq t \leq 1 | B_1 = 0\}$. It is proved as follows:

$$\begin{aligned} P(B_t \leq x | B_1 = 0) = & P(t\hat{B}_{1/t} \leq x | \hat{B}_1 = 0) \\ & P(t(\hat{B}_{1/t} - \hat{B}_1) \leq x) \end{aligned} \tag{3}$$

where $t(\hat{B}_{1/t} - \hat{B}_1) \sim N(0, t(1-t))$. So we get:

$$\begin{aligned} X_t \overset{\text{d}}{=} & \, t(\hat{B}_{1/t} - \hat{B}_1) \\ & (B_t | B_1 = 0) \end{aligned} \tag{4}$$

Thus Brownian bridge is a conditional stochastic process.

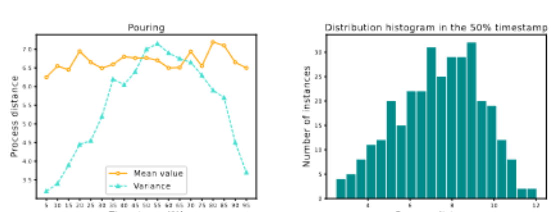
B. More Quantitative Results

B.1. Ablation Studies

Distance Measurement. We replace the distance measurement of in PCL with L_1. Table 2 shows that using L_2 as the distance measurement in PCL is more effective, and achieves better performances on various metrics.

B.2. Verification in IKEA ASM

The statistical results on representations learned by our VSP in IKEA are reported in Figure 1. The same conclusion can be drawn that it conforms to a Gaussian distribution as described in the *Equation (1)*.



(a) Distance statistic (b) Distribution on a point

Figure 1. Statistical results on the whole validation set of IKEA.

*Equal contributions.

†Corresponding author.

	Method	AP@5	AP@10	AP@15
Penn Action	SaL	76.04	75.77	75.61
	TCN	77.84	77.51	77.28
	TCC	76.74	76.27	75.88
	LAV	79.13	78.98	78.90
	CARL	92.28	92.10	91.82
	VSP	92.56	92.31	92.04
	VSP-P	**93.45**	**93.13**	**93.02**
Pouring	SaL	84.05	83.77	83.79
	TCN	83.56	83.31	83.01
	TCC	87.16	86.68	86.54
	LAV	89.13	89.13	89.22
	VSP	91.85	91.70	91.52
	VSP-P	**93.18**	**93.01**	**92.96**
IKEA ASM	SaL	15.15	14.90	14.72
	TCN	19.15	19.19	19.33
	TCC	19.80	19.64	19.68
	LAV	23.89	23.65	23.56
	VSP	26.54	26.39	26.36
	VSP-P	**28.48**	**28.27**	**28.22**

Table 1. Fine-grained frame retrieval results. **Best** and second best results are highlighted.

Distance	Classification	AP@5	Progress	τ
L_1	92.73	91.69	0.887	0.925
L_2	**93.12**	**92.56**	**0.923**	**0.986**

Table 2. Ablation studies of process distance measurement in PCL on PennAction.

B.3. Fine-Grained Frame Retrieval

This downstream task evaluates the consistency of the learned representations by the nearest neighbors. Specifically, in the validation set, we alternately take each video as a query and the others as the gallery. In each query video, we retrieve K most similar frames from the gallery for each query frame, then we get the retrieval precision for each query frame by calculating the proportion of frames belonging to the same subaction as the query frame in the K retrieved frames. At last, we report the average retrieval precision on the validation set. The quantitative results on the three datasets are shown in Table 1. We find that VSP surpasses prior methods. And training with the prompt of phase labels improves the overall performance of the three datasets.

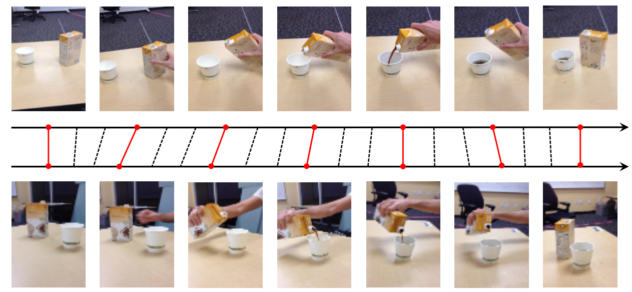


Figure 2. Visualization of video alignment on multiple-view of pouring. The two-row pictures are the temporally corresponding frames of the view pair. The slope of lines between the two timelines indicates the timestamp distance of the aligned frame pair, *e.g.*, the vertical line means their timestamp distance is 0.

Figure 3. Visualization of fine-grained frame retrieval. The leftmost column is the query and the right 5 columns are the Top-5 retrieved frames. The query action of the first row is *Baseball Pitch*. The rest of the three queries come from three different phases of *Clean and jerk*.

C. More Qualitative Results

C.1. Video Alignment

We evaluate our VSP on another downstream task, video alignment. Video alignment aims to find the temporal correspondence between two videos of the same action. In our experiments, we randomly select video pairs from Pouring that provide multiple-view of the same action and object. Then we extract their frame-wise representations with our framework and calculate their cosine similarities, which we use to find the temporal correspondence via the dynamic time warping (DTW) algorithm [1, 2]. A visualized result from the Pouring test set is shown in Figure 2. We can find that corresponding frames have the similar semantic, *i.e.*, the same action phase, which suggests that our method can grab motion dynamics to align videos temporally.

C.2. Fine-grained Frame Retrieval

We show the visualization of fine-grained frame retrieval on PennAction [3]. We first extract the representations of each video in the validation set. Then we randomly select one video for query and the other videos as the gallery. For each query frame of the query video, we retrieve its K nearest neighbors in the embedding space from the gallery. Figure 3 shows examples of Top-5 retrieval results. Most of the retrieved frames belong to the same subaction as the query frame. The semantics of motion is more important than backgrounds or camera views, which illustrates our method is sensitive to motion dynamics.

D. Method Algorithm

Brownian bridge length η and overlap ratio δ. As the Brownian bridge represents a subaction, an intuitive way to get rid of η is using the average length (denote as l) of the subaction in the target dataset. Overlap ratio δ should simultaneously maintain continuity and discrimination of consecutive subactions. 10%-30% is a safe range according to the experiments.

Pseudo-code. To further elaborate our method, we present the pseudo-code in the Algorithm 1 in Pytorch style. We list the variables used in the algorithm at the top. Corresponding to *Section 3*, we first encode the given frame triples into embeddings. Then we show the algorithm for Brownian bridge construction under various annotation situations. At last, we detail the process contrastive training, which contains three parts: **a)** bridge distance, **b)** process-based contrastive loss, and **c)** supervised contrastive loss. Based on the algorithm, we can find that our method are model-agnostic and simple yet efficient.

References

[1] Rémi Lajugie, Damien Garreau, Francis R. Bach, and Sylvain Arlot. Metric learning for temporal sequence alignment. In *NeurIPS*, 2014. 2

[2] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:159–165, 1978. 2

[3] Weiyu Zhang, Menglong Zhu, and Konstantinos G. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. *ICCV*, 2013. 3

Algorithm 1 Pytorch-style pseudo-code of Video as Stochastic Process (VSP) Framework

```
# [Parameters] f_theta: encoder network
# [Arguments ] N: batch size, T: video length, D: embedding size, eta: bridge length, delta: overlap ratio
# [Inputs    ] batch: a batch with N videos

#===========================
# Section 3.1: Video Encoding
#===========================
# VSP is model-agnostic
Z = f_theta(batch) # NxTxD

#================================
# Section 3.2: Bridge Construction
#================================
for i in range(N):
    if annotation_type == 'raw_videos':
        # if no phase annotation available, random sample clips according to eta, delta
        clips = range(start=1, stop=T, step=eta, overlap=delta)
        sample_clip = clips.random()
    else:
        # else random sample phases
        sample_clip = phase_annotation.random()

    X_1[i],X_T[i] = sample_clip[0], sample_clip[-1] # X indicates timestamps
    X_t[i] = random(x_1[i], X_T[i]) # sample internal frame timestamp

    # retrieve embedding features
    Z_1[i], Z_t[i], Z_T[i]  = Z[i, X_1[i]], Z[i, X_t[i]], Z[i, X_t[i]] # NxD

#=========================================
# Section 3.3: Process Contrastive Training
#=========================================
# a) bridge distance, i.e., Eqn.(2)
def distance(z_1, z_t, z_T, x_1, x_t, x_T):
    sigma = (x_t-x_1) * (x_T-x_t) / (x_T-x_1)  # x_t-x_1=t,x_T-x_1=T
    zt_ = z_1 * (x_T-x_t)/(x_T-x_1) + z_T * (x_t-x_1)/(x_T-x_1)
    distance = -1/(2*sigma.pow(2)) * euclidean_dist(zt,zt_).pow(2)
    return distance

# b) process-based contrastive loss, i.e., Eqn.(3)
def pcl(pos_dis, neg_dis):
    ps = torch.exp(pos_dis)
    ns = sum(torch.exp(neg_dis))
    loss = -torch.log(ps/(ps+ns))
    return loss

# c) supervised contrastive loss, i.e., Eqn.(4)
def scl(target, positives, negatives):
    ps = sum(torch.exp(dot(target, positives)/tao))
    ns = sum(torch.exp(dot(target, negatives)/tao))
    loss = -torch.log(ps/(ps+ns))
    return loss

# loop for all videos
for i in range(batch):
    pos_dis[i] = distance(Z_1[i], Z_t[i], Z_T[i], X_1[i], X_t[i], X_T[i])

    # negative samples are all sampled timestamps except for positive, i.e., each positive has (N-1)*3 negative
    batch_X = X_1.pop(i)+X_t.pop(i)+X_T.pop(i) # i.e., \mathcal{B}
    neg_dis[i] = [distance(Z_1[i], Z_t[neg_x], Z_T[i], X_1[i], X_t[neg_x], x_T[i]) for neg_x in batch_X]

    # process-based contrastive loss
    loss += pcl(pos_dis[i], neg_dis[i])

    # if we have frame-level annotations, we can further distinguish pos/neg timestamps as supervisions
    if annotation_type == 'frame_labels':
        pos_batch_X = [x for x in batch_X if frame_annotation[x] == frame_annotation[X_t[i]]] # i.e., \mathcal{P}
        neg_batch_X = batch_X - pos_batch_X # i.e., \mathcal{N}

        # supervised contrastive loss
        loss += scl(Z_t[i], pos_batch_X, neg_batch_X)

# optimization step
loss.backward()
optimizer.step()
```