Context-aware and Scale-insensitive Temporal Repetition Counting

Huaidong Zhang[1], Xuemiao Xu [*1,2,3], Guoqiang Han[1], and Shengfeng He[* 1]

[1]South China University of Technology
[2]State Key Laboratory of Subtropical Building Science
[3]Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information

(a) Short Cycle (Jumping Rope)   (b) Long Cycle (Bench Press)

(c) Varied Cycle (Playing Violin)   (d) Double-Motion Cycle (Front Crawl)
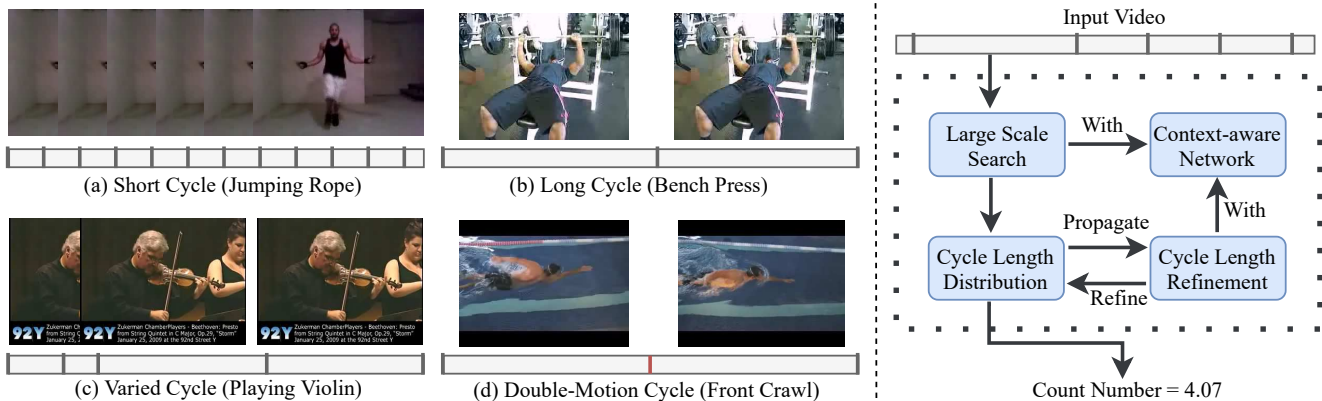
Count Number = 4.07

Figure 1: The varied cycle lengths and context-dependent motions of repetitive actions pose challenges for counting temporal repetitions. We propose a context-aware and scale-insensitive framework to cope with these problems. The counting process is designed in a coarse-to-fine manner, integrating with a context-aware network for detecting bidirectional repetitive actions.

Abstract

Temporal repetition counting aims to estimate the number of cycles of a given repetitive action. Existing deep learning methods assume repetitive actions are performed in a fixed time-scale, which is invalid for the complex repetitive actions in real life. In this paper, we tailor a context-aware and scale-insensitive framework, to tackle the challenges in repetition counting caused by the unknown and diverse cycle-lengths. Our approach combines two key insights: (1) Cycle lengths from different actions are unpredictable that require large-scale searching, but, once a coarse cycle length is determined, the variety between repetitions can be overcome by regression. (2) Determining the cycle length cannot only rely on a short fragment of video but a contextual understanding. The first point is implemented by a coarse-to-fine cycle refinement method. It avoids the heavy computation of exhaustively searching all the cycle lengths in the video, and, instead, it propagates the coarse prediction for further refinement in a hierarchical manner. We secondly propose a bidirectional cycle length estimation method for a context-aware prediction. It is a regression network that takes two consecutive coarse cycles as input, and predicts the locations of the previous and next repetitive cycles. To benefit the training and evaluation of temporal repetition counting area, we construct a new and largest benchmark, which contains 526 videos with diverse repetitive actions. Extensive experiments show that the proposed network trained on a single dataset outperforms state-of-the-art methods on several benchmarks, indicating that the proposed framework is general enough to capture repetition patterns across domains. Code and data are available in `https://github.com/Xiaodomgdomg/Deep-Temporal-Repetition-Counting`*.*

1. Introduction

Human activities are commonly involved repetitive actions. Temporal repetition counting is a problem that aims to count the number of repetitive actions in a video [7, 14, 21, 26]. The repetition analysis is explored as an auxiliary cue to other video analysis applications, such as cardiac and respiratory signal recover [16], pedestrian detection [22], 3D reconstruction [15, 24], and camera calibration [11].

This is a challenging problem as repetitive actions exhibit inherently different action patterns. We summarize 4 representative cases in the left part of Figure 1. Figure 1(a) and (b) show the most common repetitions, in which actions are performed in fixed cycles. The problem of detecting these two repetitions is that their cycle lengths varied largely, and therefore is invalid to make restricted assumptions about the time-scale of the cycle length across actions.

---

*Xuemiao Xu and Shengfeng He are joint corresponding authors.
Email: xuemx@scut.edu.cn, hesfe@scut.edu.cn

In Figure 1(c), the case of playing the violin shows that the cycle lengths are not always a fixed value. This case is contradictory to (a) and (b), and hence the assumption of actions will be performed in a periodic manner is false. In Figure 1(d), a front crawl action can be decomposed into two sub-actions with a similar motion field, crawling with the left hand and right hand. As the two sub-actions are similar in motion space, contextual information in semantic space should be considered to avoid the double counting error.

Most existing methods [3, 7, 14, 16, 21] rely heavily on the periodicity assumption. As a consequence, although the representative work [14] achieves a near-perfect performance on the periodic dataset *YTsegments*, it cannot detect varied cycle lengths in the non-stationary video dataset *QUVA Repetition* [25]. While the latest work [26] address this problem, it detect repetition solely based on the motion field. Therefore it conflicts with the scenarios like Figure 1(d), in which repetitions cannot be distinguished by motion field and contextual and semantic information is required to understand the action. Based on the above observations, we argue that detecting repetitions should 1) exhaustive search for a large range of cycle lengths to cover most unknown actions; 2) include contextual understanding and estimating cycle lengths by taking multiple periods into consideration.

In this paper, we tailor a context-aware and scale-insensitive framework based on the above principles. The data flow is shown in the right part of Figure 1. Following rule #1 to exhaustively search all the time scales can absolutely address the cycle lengths variations problem, but it leads to expensive computation. We combat this problem by proposing a coarse-to-fine cycle lengths estimation strategy integrated with a regression network. In particular, we only exhaustive search the initial cycle lengths for a local video clip. The initial estimation, is then propagated to the entire video, and each of the estimated repetition in the video is refined by our regression model. In this way, we largely reduce the computational cost in searching accurate cycle lengths, while we can adapt to large variations of cycle lengths in the same video. The proposed regression model handles rule #2, in which we inject contextual information for estimating accurate cycle lengths. Specifically, instead of taking only one action cycle as input, we sample the video to contain two consecutive repetitions, named double-cycle. Given such broad context, our regression model aims to relocate the previous and future repetitive cycles in a bidirectional manner. Furthermore, existing researches in repetition counting lack of sufficient data, therefore we propose a new repetitive action counting benchmark, named *UCFRep*. It is constructed by annotating repetitive actions from the widely used dataset *UCF101* [28], and it is the largest dataset containing 526 videos. Extensive experi-

ments demonstrate the proposed method is able to cope with various repetitive actions, and we outperform state-of-the-art methods on three benchmarks.

Our contributions are four-fold:

(1) We propose a coarse-to-fine double-cycle estimation strategy integrated with regression, which allows fast estimation of cycle lengths for the entire video and dynamic relocation of varied cycles.

(2) We present a bidirectional context-aware regression model. It explores contextual information to simultaneously estimate the previous and future cycles in a bidirectional manner.

(3) We construct a new and largest benchmark *UCFRep*. 526 repetitive action videos are annotated for training and evaluation.

(4) The proposed network outperforms state-of-the-art methods on three benchmarks, especially we achieve superior performances on two unseen benchmarks (without fine-tuning). It reveals the proposed framework is general enough to complex and unknown scenes.

2. Related Work

A typical solution for temporal repetition counting is to transfer the motion field into one-dimensional signals, and then they try to recover the repetition structure from the signal period [1, 13, 19, 20, 30]. The mainstream of these methods obtains repetition frequency with Fourier analysis [2, 3, 7, 21]. In addition, they detect the cycle by filtering [4], peak detection [29], classification [8], and singular value decomposition [6]. The above methods assume that the estimating repetition is periodic, so that they cannot handle the non-stationary repetitions. A recent work [26] addresses this limitation, and propose a novel inference scheme to detect non-stationary actions. However, they only adopt the motion field to extract features for analysis, while ignoring context-dependency in semantic domain.

Like us, there are methods that also use deep features for repetition analysis. Li *et al.* [16] propose to learn temporal dependency by adopting the LSTM network on the sequence of images. They aim to recover the cardiac and respiratory signals from the medical image sequence, as such their method cannot handle complex repetitions in real-world. Levy and Wolf [14] aim to propose a classification network for live repetition estimation. Their network is designed to extract features of 20 frames from the video with the predefined sampling-rate. As discussed above, a predefined cycle lengths cannot adapt to the complex repetitive actions with large variations of cycle lengths.

Action localization [17, 18, 27] shares a similar spirit to localize actions in temporal domain. These methods aim
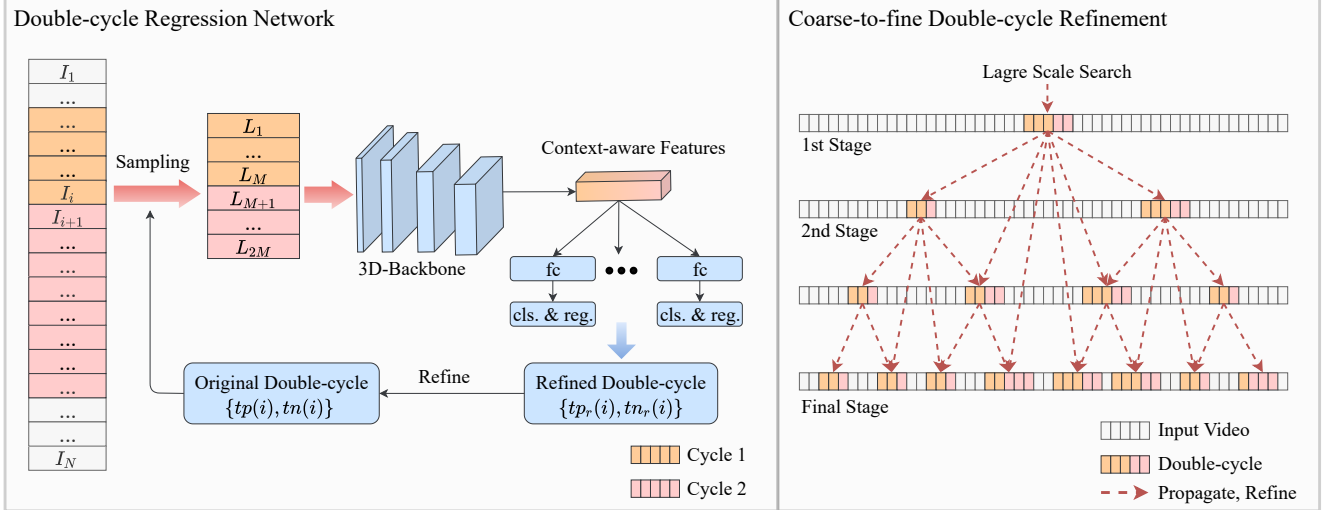
Figure 2: Framework overview. The proposed context-aware double-cycle regression network is shown on the left. It regresses a new double-cycle $\{tp_r(i), tn_r(i)\}$ based on the context information sampled from the previous double-cycle $\{tp(i), tn(i)\}$. In the right part, a coarse-to-fine double-cycle refinement method is illustrated. We first perform exhaustive search locally on the first stage, and the initial double-cycle is propagated and refined in the following stages. An accurate counting result can be obtained by averaging all the cycle lengths in the video.

to locate the temporal begin and ending points of each action in the entire video, hence these methods can be easily adapted to the field of repetition counting. However, these methods find the action segment separately, which means that they ignore the repetition priors to effectively utilize the context information. In our method, we borrow the idea of the anchor-based temporal regression from this literature, and further explore context dependency.

3. Approach

In this section, we first introduce the problem formulation and overview of the proposed context-aware and scale-insensitive framework. Then we describe two core modules of our framework, the context-aware double-cycle regression network and coarse-to-fine double-cycle refinement. Finally, we present the details of our newly constructed temporal repetition benchmark.

3.1. Problem Formulation

Repetition definition. We have a different problem setting than prior works, as we aim to locate both previous and future cycles in a bidirectional manner. Given a video with N frames $\mathcal{I} = \{I_1, I_2, ..., I_N\}$, the repetition can be defined as follows: for a frame I_i, if we can find a previous frame $I_{p(i)}$ and a future frame $I_{n(i)}$, such that the two frame sequences $\{I_{p(i)}, I_{p(i)+1}, ..., I_i\}$ and $\{I_i, I_{i+1}, ..., I_{n(i)}\}$ contains the identical actions, then there are two repetitions existing in these two sequences. We refer these two consecutive cycles as *double-cycle*, and the $I_{p(i)}$ as the *previous repetitive frame* of I_i and $I_{n(i)}$ as the *next repetitive frame* of I_i.

Target formulation. In this paper, we aim to count the temporal repetition number c for the given video. If the action is strongly periodic in the video, we can assume the cycle length is a constant across the entire video. Then we can easily estimate the repetitions number by finding the previous and next repetitive frame locations $\{p(i), n(i)\}$ of an arbitrary frame i and calculate the number of repetitions c as:

$$c = \frac{N}{i - p(i) + 1} = \frac{N}{n(i) - i + 1}. \tag{1}$$

However, the variety between repetitions cannot be neglected in the real-world. To tackle this problem, we propose to calculate the repetition counts by estimating $\{p(i), n(i)\}$ of each frame in the video. Therefore we formulate the problem as

$$c = \sum_{i=1}^{N} \left(\frac{0.5}{i - p(i) + 1} + \frac{0.5}{n(i) - i + 1} \right). \tag{2}$$

Two cycle lengths can be computed as $tp(i) = i - p(i) + 1$ and $tn(i) = n(i) - i + 1$. For clarity, we define $\{tp(i), tn(i)\}$ as the *double-cycle* that describes two consecutive repetitions with frame i.

3.2. Framework Overview

Following the target formulation, our framework is designed to predict the double-cycle $\{tp(i), tn(i)\}$ for all the position $i \in \{1, 2, ..., N\}$. We first propose a context-aware double-cycle regression network, which is illustrated in the left part of Figure 2 and described in Section 3.3. The network is designed to refine the given double-cycle for a

specific position. Given an initial double-cycle, our network extracts the 3D features based on some sampled video frames and outputs a new double-cycle $\{tp_r(i), tn_r(i)\}$. With the extracted context information from a large range of video frames, the network is able to identify the repetition and regress the cycle lengths easily. Furthermore, this process is performed multiple times to obtain a progressively refined double-cycle.

As discussed above, an exhaustive search should be performed to cope with the large cycle length variation problem. It can also provide an reasonable initial double-cycle for the regression network. Instead of searching the entire video, we first search locally in the video, and propagated the prediction to the other frames. The right part of Figure 2 shows our method and it is described in Section 3.4. We perform exhaustive searching for one time in the middle frame of the video, such that the initial double-cycle is likely within the same scale with others. It is then propagated to the other frames, each of the new frame is integrated with the regression network for local refinement. For each stage we sample the positions uniformly across the video so that the sampled position can be the propagation root for the next stages. The final repetition counts of the video can be calculated by the repetition count summarization of all frames.

3.3. Double-cycle Regression Network

The objective of the network is to refine the input double-cycle $\{tp_r(i), tn_r(i)\}$ of an assigned position i. To extract features of fixed size for regression, we sample specific frames within the double-cycle. As illustrated in the left part of Figure 2, network input L is a sequence with $2M$ frames, which consists of two half. We sample the first half of the inputs uniformly from the range $[i - 2tp(i), i]$, and the next half inputs from the range $[i + 1, i + 2tn(i) + 1]$. Note that we double the sampling range to detect large context like the double-motion in Figure 1(d). The sampled sequence L is then fed into a 3D-backbone model. We use the 3D-ResNext101 [10, 31] pretrained on the ActivityNet [5]. Other network architectures are also applied, please refer to the experiments for details. We remove the last classification layer and use the outputs after pooling to be the context-aware 1D-features (4096 dimensions for ResNext101). The features are then fed into the newly added prediction branch for classification and regression. The prediction branch is a two fully-connected layers with multi-anchor, where we use 7 anchors with default size $\{0.5, 0.66, 0.8, 1.0, 1.25, 1.5, 2.0\}$ to detect different size of the repetition. Note that totally 14 anchors are used since we have two cycles $\{tp_r(i), tn_r(i)\}$.

During training, the 3D backbone and the added branch are trained end-to-end with classification loss and regression loss. With the network outputs for classification

---

Algorithm 1 Coarse-to-fine Double-cycle Refinement

Input: Video length N, double-cycle regression network F, number of refinement stages K
Output: Double-cycle prediction $\{tp, tn\}$
1: Initialize queue Q
2: Determine $\{tp(N/2), tn(N/2)\}$ by global search with network F
3: Push $\{tp(N/2), tn(N/2)\}$ into Q
4: **for** $k = 1, 2, ..., $ K-1 **do**
5: Initialize point set S_k with 2^k points sampled uniformly over video
6: Initialize $\{tp(S_k), tn(S_k)\}$ with the prediction in Q
7: Iteratively refine $\{tp(S_k), tn(S_k)\}$ with network F
8: Push $\{tp(S_k), tn(S_k)\}$ into Q
9: **end for**
10: **return** $\{tp, tn\}$ in queue Q

$\{y_p, y_n\}$ and for regression $\{t_p, t_n\}$, we formulate the overall loss function:

$$\mathcal{L} = (\mathcal{L}_{cls}(y_p, \tilde{y_p}) + \mathcal{L}_{cls}(y_n, \tilde{y_n})) + \lambda \left(\mathcal{L}_{reg}(t_p, \tilde{t_p}) + \mathcal{L}_{reg}(t_n, \tilde{t_n}) \right), \tag{3}$$

where $\mathcal{L}_{cls}$ is the cross-entropy loss after softmax and $\mathcal{L}_{reg}$ is the smooth L_1 regression loss [23]. $\{\tilde{t_p}, \tilde{t_n}\}$ is the repetition ground truth with the parameterizations of scale-invariant center translation and the log-space cycle-lengths shifting [9]. $\{\tilde{y_p}, \tilde{y_n}\}$ is the classification label that equals to 1 if intersection-over-union (IoU) of double-cycle prediction and ground truth is greater than 0.5, and 0 otherwise. λ is the weighting factor that empirically set to 50. During inference, the objective $\{tp_r(i), tn_r(i)\}$ is equal to the regression output of the anchor which has the highest classification score.

3.4. Coarse-to-Fine Double-cycle Refinement

Since the network extracts the features from the context determined by the original double-cycle $\{tp(i), tn(i)\}$, a good initialization will be helpful to improve localization. To this end, we propose a hierarchical pipeline to provide initialization by determining the double-cycle in a coarse-to-fine manner. The key idea of the proposed pipeline is the cycle length variation between different frames can be overcome by regression, especially for the neighboring frames. Therefore each stage we refine the results on the uniformly sampled positions across the video, so that the initialization of the next stage can benefit from the neighboring prediction of the previous stage. As illustrated in the right part of Figure 2, in the kth stage, we predict $\{tp(i), tn(i)\}$ on the uniformly sampled position $i = \{N/2^k, 3N/2^k, ..., (2k - 1)N/2^k\}$. The prediction for each position consists of two process, the initialization and refinement. Algorithm 1 illustrates the initialization and refinement pipeline.
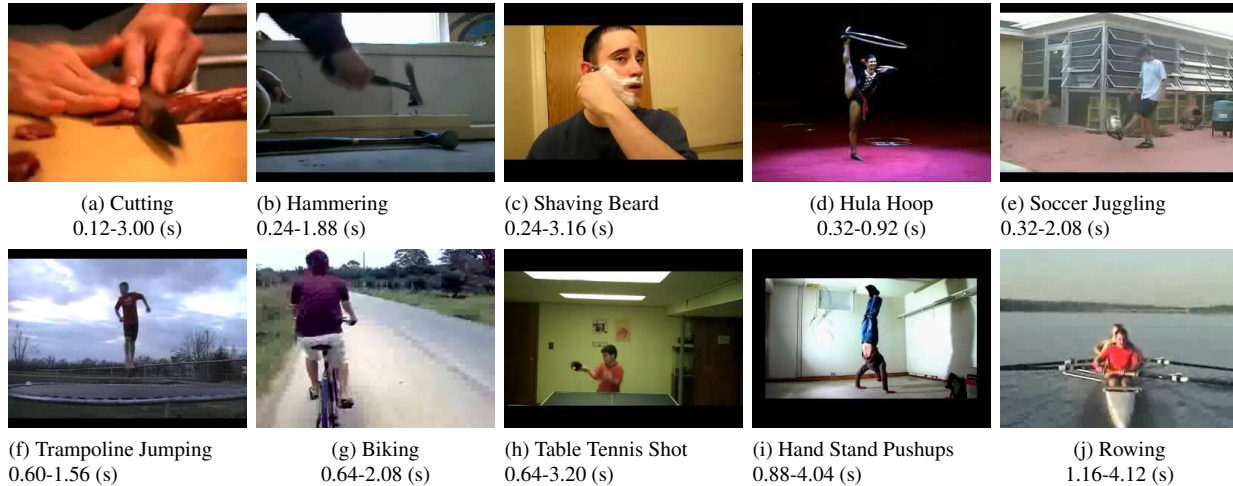
Figure 3: 10 examples from different categories of the *UCFRep* benchmark. We annotate the minimum and maximum cycle-length of each category below the image, indicating the cycle-length variation.

Initialization. For the first stage, we let the double-cycle of the middle position, $\{tp(N/2), tn(N/2)\}$, equal to the value sampled from the large scale $[\mu_1, N/\mu_2]$, and then determine the initialized scale by the network classification confidence. In the other stages, we propagate the prediction from the previous stage as initialization, following the arrow direction in the right part of Figure 2. In particular, each position finds the previous refined neighbors for initialization. If only one neighbor is available (the first/last position of the current stage), we use it as the initialization directly. Otherwise, we merge the two observations from the previous neighbor and next neighbor averagely. Under this scheme, we do the heavy computation search only one time in the first stage, and effectively utilize the refined results for the initialization of all the frames.

Refinement. After initialization, we refine the double-cycle estimation for the given position i. With the refined results $\{tp_r(i), tn_r(i)\}$ from the regression network, we update the observation on position i with the exponential moving average mechanism. In other word, we update the estimation with the equation $\{tp(i), tn(i)\} = \beta\{tp(i), tn(i)\} + (1 - \beta)\{tp_r(i), tn_r(i)\}$, where β is the decay factor set as 0.5 empirically. Note that the refinement can be performed iteratively to achieve more precise results.

After the coarse-to-fine refinement, we obtain the cycle length prediction on uniformly sampled positions. To count the action by sampling 2^{K-1} points rather by all the N frames, we use the prediction of the final stage to present the prediction of all the frames by modifying Equation 2:

$$c = \sum_{i=1}^{2^{K-1}} \frac{N}{2^{K-1}} \left(\frac{0.5}{tp(s)} + \frac{0.5}{tn(s)} \right), s = \left\lfloor \frac{(2i-1)N}{2^K} \right\rfloor,$$
(4)

where Kth stage is the final stage.

3.5. UCFRep Benchmark

The previous repetition datasets *YTsegments* [14] and *QUVA Repetition* [25] contain only 100 videos for evaluation. Due to the lack of labeled data, the previous deep learning work [14] trains their model on synthesis data. Despite the tailored design of the simulation, the domain gap between synthesis data and real data is unneglectable. Motivated by this, we present an action repetition dataset, called *UCFRep* benchmark, aiming to provide an environment for training and evaluate the data-driven model. All the data in the proposed benchmark are collected from the widely used action recognition dataset *UCF101* [28]. Therefore, the proposed benchmark focuses on evaluating the repetition counting performance of human action. Despite all the data is labeled with category, we find that the proposed network trained on the benchmark is general enough to perform well on the previous unseen dataset *YTsegments* and *QUVA Repetition* in experiments. We mainly introduce the benchmark from three aspects, data collection, repetition labeling, and dataset statistic.

Data collection. The original *UCF101* [28] is an action recognition data set of action videos. 13320 videos are collected from YouTube and further classified into 101 action categories. Videos in each category are grouped into 25 groups according to whether they share common features, such as similar backgrounds, viewpoints, etc. We check all the 101 categories from the dataset and select 23 categories in which the action is taken cyclically. Examples of 10 categories are shown in Figure 3.

Repetition labeling. We annotate the temporal bound of repetitions similar to the principle in *QUVA Repetition* [25]. Two human annotators are invited to mark out the interval contain repetitions and the repetitive frames in each video. First, from each group in the original *UCF101*, we ask the

	YTSeg	QUVA	Ours
Num. of Videos	100	100	**526**
Duration(s)	1487	1754	**3500**
Num. of Counts	1080	1246	**3506**
Count Min/Max	4/51	4/63	3/54
Min of Cycle(s)	-	0.20	0.12
Max of Cycle(s)	-	7.69	6.76
Max/Min of Cycle	-	38.76	**56.33**
Cycle Variation	0.22	0.36	**0.42**

Table 1: Dataset statistic of *YTsegments* [14], *QUVA Repetition* [25] and the proposed *UCFRep*. Our dataset is larger than the previous datasets in terms of the number of videos, total duration and number of annotations. The wide range of cycle length between videos and large variation within the video also indicate that our benchmark is more challenges. The cycle variation is the average value of the absolute difference between minimum and maximum cycle length divided by the average cycle length.

annotators to choose one video with the clearest repetitions. If no repetitions can be founded, all the videos in this group will be abandoned. As a result, 49 groups cannot find any repetition and $23 \cdot 25 - 49 = 526$ videos are collected in our benchmark. With these videos, we let the annotators determine the repetition interval. We consider the first frame of the interval as the reference, and ask the annotators to mark all the repetitive frames of reference within the interval. Finally we use the average value of their annotations as the final label, and the number of repetitive frames determine the repetition counts.

Dataset statistic. We summarize the dataset statistic in Table 1. In the proposed benchmark, we provide totally 526 videos containing 3500 seconds. 3506 cycle bounds are annotated in our benchmark to provide abundant data for training and evaluation. The benchmark also has a larger variation compared with the previous datasets. The Max/min of Cycle indicates the difficulty from the diverse time-scale between different types of the repetitions, and the cycle variation shows the cycle-length variation within the video.

4. Experiments

Implementation Details. We implement the proposed network using Pytorch, and test it with an NVIDIA Geforce GTX1080Ti GPU. All input video frames of the network are resized to 112×112, and we construct a $2M = 32$ frames sequences. For training, we use Adam optimizer [12] with a fixed learning rate of 0.00005 and batch size of 24. We train our network on the *UCFRep* with 100 epochs.

We train our network with the same pipeline of the proposed coarse-to-fine refinement. Data augmentation is used to extend the annotations: if the variation of two consecutive repetitions is less than 0.3, we assume they are periodic. Then we add annotations within the interval automatically

by linear interpolation.

During testing, we perform the coarse-to-fine refinement with $K = 5$ stages. Our initial exhaustive searching is performed with 30 scales (ranging from 4 to $N/2$), and conduct 4 times refinement in the 1st and 2nd stages, 2 times in the 3rd stage, and 1 time in the 4th to 5th stages, leading to $30 + 4 \cdot (1 + 2) + 2 \cdot 4 + 8 + 16 = 74$ forwards of the estimation network. The running time of our method depends on the times of the network forwards, and it takes averagely 1.8 seconds to process a video.

Evaluation Datasets. We evaluate our method on the three video datasets: the existing datasets *YTsegments* [14] and *QUVA Repetition* [25], as well as the proposed benchmark *UCFRep*. Both the *YTsegments* and *QUVA Repetition* contain 100 videos with a wide range of repetitions, like sports of humans and animal behaviors. We consider all the videos from *YTsegments* dataset and *QUVA Repetition* dataset as testing set, and all the training and the validation is done on the proposed *UCFRep* benchmark. As a result, we split the videos in *UCFRep* benchmark into the training set and validation set according to the group number from *UCF101*. 421 videos with group numbers 1-20 are split into the training set, and 105 videos with group numbers 21-25 are in the validation set.

Evaluation Metric. Following the previous works [14, 26], we evaluate the proposed method by counting accuracy. For each dataset, we report the mean absolute error (MAE) and off-by-one-accuracy (OBOA) given K videos

$$\text{MAE} = \frac{1}{K} \sum_{i=1}^{K} \frac{|\tilde{c}_i - c_i|}{\tilde{c}_i}, \tag{5}$$

$$\text{OBOA} = \frac{1}{K} \sum_{i=1}^{K} [|\tilde{c}_i - c_i| \le 1], \tag{6}$$

where $\tilde{c}$ is the ground truth repetition counts. The mean absolute error is a widely used metric to directly evaluate counting errors. The off-by-one-accuracy can counts the rounding error and show the possible cycle cut-offs at both ends of the video as introduced in [26].

4.1. Comparison with Other Methods

The comparison with the existing methods for temporal repetition counting is shown in Table 2. We compare our method with two hand-crafted feature methods [21, 26] and one deep learning-based method [14]. As the complete source codes of [21, 26] are unavailable, we compare to them on two previous testing datasets *QUVA Repetition* and *YTsegments*. We can observe that our method can outperform all the previous methods. It demonstrates that our method trained on the *UCFRep* is general enough

Method	QUVA Repetition [25]		YTsegments [14]		UCFRep (Ours)	
	MAE↓	OBOA↑	MAE↓	OBOA↑	MAE↓	OBOA↑
Pogalin *et al.* [21]	0.385 ± 0.376	0.49	0.219 ± 0.301	0.68	-	-
Levy and Wolf [14]	0.482 ± 0.615	0.45	0.065 ± 0.092	0.90	-	-
Levy and Wolf* [14]	0.237 ± 0.339	0.52	0.142 ± 0.231	0.73	0.286 ± 0.574	0.68
Runia *et al.* [25]	0.232 ± 0.344	0.62	0.103 ± 0.198	0.89	-	-
Runia *et al.* [26]	0.261 ± 0.396	0.62	0.094 ± 0.174	0.89	-	-
Ours-Resnet18	0.190 ± 0.327	0.70	0.062 ± 0.125	0.91	0.213 ± 0.343	0.69
Ours-Resnet50	0.167 ± 0.293	0.75	0.081 ± 0.261	0.94	0.190 ± 0.288	0.74
Ours-Resnet101	$\mathbf{0.148 \pm 0.290}$	0.75	0.066 ± 0.170	0.94	0.187 ± 0.303	0.77
Ours-Resnext101	0.163 ± 0.311	**0.76**	$\mathbf{0.053 \pm 0.115}$	0.95	$\mathbf{0.147 \pm 0.243}$	0.79

Table 2: Comparison with the existing methods on *YTsegments*, *QUVA Repetition* and *UCFRep* for temporal repetition counting. The method with * is the re-implementation version by us trained on our *UCFRep* benchmark.
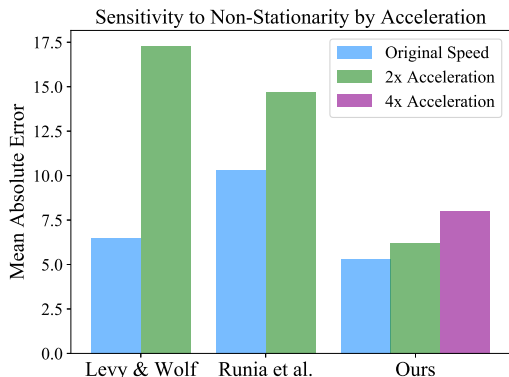

Figure 4: Evaluation on the robustness to acceleration 1x, 2x and 4x on the *YTsegments* dataset following the experiments in [25]. Compared to a previous scale-insensitive method [25], our method is more robust to the time-scale.

	MAE↓	OBOA↑	Iterations
Stage 3	0.157 ± 0.284	0.78	50
Stage 4	0.156 ± 0.254	0.78	58
Stage 5	$\mathbf{0.147 \pm 0.243}$	**0.79**	74
Stage 6	0.151 ± 0.254	**0.79**	106

Table 3: Ablation study of the proposed coarse-to-fine refinement method on the *UCFRep* benchmark validation set.

	MAE↓	OBOA↑
Fixed	0.177 ± 0.280	0.70
Fixed+mAnchor	0.171 ± 0.249	0.71
Free	0.157 ± 0.243	0.76
Free+mAnchor	$\mathbf{0.147 \pm 0.243}$	**0.79**

Table 4: Ablation study of the proposed context-aware estimation network on the *UCFRep* benchmark validation set.

to the common repetitions from other datasets. Especially for the non-stationary dataset *QUVA Repetition*, our method obtains improvement on MAE with 6.9% and OBOA with 14%, indicating that our scale-insensitive framework can better handle the videos with varied cycle-length.

To demonstrate these improvements are brought mainly by the proposed framework rather than the new dataset, we fine-tune the learning-based method [14] on the new benchmark using our train/validation protocol. Note that the other two competitors [21, 26] are training-free methods. The original implementation [14] uses a simple 3D network to learn on synthesis data with 20 50×50 images as input. We replace their network with Resnext101 to extract information from 32 112×112 frames for adapting to the higher-dimensional data. We remove their ROI detection to keep the inference sequence similar to the training data, and the other implementations follow the published official code. Not surprisingly, because of the increased number of training data, the re-trained model on *UCFRep* benchmark shows better performance compared with the original implementation on the *QUVA Repetition* dataset. However, it cannot perform well on the periodic dataset *YTsegments*,

this is because their synthesis data is created following the restrict periodic assumption, while our dataset shows various types of repetitions. Compared with both the finetuned and original versions, our method outperforms them on all the datasets, as their network is designed to consider only a fixed scale of action. These results also demonstrate the success of our tailored context-aware and scale-insensitive framework.

We further evaluate the robustness to time-scale of our method. We follow [25] to manually speed up the video to achieve different time-scales. As shown in Figure 4, when the video is processed with different speeds, it poses a challenge to the fixed time-scale method [14] (6.5% on 1x and 17.3% on 2x). Compared with the results (10.3% on 1x and 14.7% on 2x) from the existing scale-insensitive method [25], our method is more robust to speed variations (5.3% on 1x, 6.2% on 2x and 8.0% on 4x), which implies that our method can detect the repetitions with different time-scales.

4.2. Ablation Study

We conduct the ablation study on *UCFRep* validation set. In Table 3, we compare the performance of our system uti-

1st Stage: 9 9 9 9 9 9 9 9 9 9 9
3rd Stage: 13 13 13 14 14 9 9 9 13 13 13
5th Stage: 12 12 14 15 13 10 9 8 10 13 14
Ground Truths: 12 12 13 21 12 10 9 7 11 13 14

Figure 5: Multi-stage cycle length visualization of a video from the *QUVA Repetition* dataset. In this video, a man is painting (unseen during training), our coarse-to-fine strategy can progressively refine the cycle lengths.

Metric	All	HulaHoop	Biking	Hammering	Soccer
MAE-avg	0.147	0.120	0.123	0.154	0.168
MAE-std	0.243	0.240	0.062	0.170	0.111

Table 5: Performance variations with respect to different action classes on the *UCFRep* benchmark validation set.

lizing different stages as the final stage in the coarse-to-fine refinement. The process with 6 stages will involve 32 iterations in the final stage, thus it overall needs $74 + 32 = 106$ iterations. The results in this table indicate that involving more stages and computations in the refinement process can improve the results. We balance the trade-off between accuracy and speed, and choose stage 5 as the final stage.

We also compare the performance of our context-aware network with the other network designs in Table 4. We first compare the performance of using double time-scales for the two consecutive repetitions (Free) or single time-scale shared by the consecutive repetitions (Fixed). The results with double time-scales are better than those with a single time-scale, which demonstrates that the free time-scales help to tackle the diverse cycle length. In addition, the multi-anchors design (mAnchor) achieves the best performance integrated with the double time-scales. This implies that the regression can refine the cycle length with a large range, and thus benefitted from the multi-anchors prediction focusing on the diverse time-scales.

In Table 5, we further show the performance variations with respect to different action classes. We can see that the variations within the same action class are relatively small, indicating that our model is instance and class insensitive.

4.3. Refinement Results Visualization

To show the process of coarse-to-fine refinement, we visualize the prediction of the 1st stage, 3rd stage and the 5th stage over a video from *QUVA Repetition* dataset in Figure 5. We set the each repetition prediction equal to the rounded mean value of the cycle length from the closet sampled position. From the results, we can find that we give an identical estimation to all the positions in stage 1 since it only involves one local prediction. In the 3rd stage and 5th stage, the predictions after propagation and refinement achieve high overlap with the ground truth, showing that the proposed coarse-to-fine refinement can overcome the variation between consecutive repetitions.

5. Conclusion

In this paper, we present a novel context-aware and scale-insensitive framework for temporal repetition counting. To tackle the challenges posed by the diverse cycle-lengths between videos and within repetitions, we propose a coarse-to-fine cycle refinement scheme. Instead of detecting the repetition with fixed time-scales, we search the time-scale with a wide range locally at the beginning and refine the scales for each temporal location in a coarse-to-fine manner. We further propose a context-aware regression network to learn contextual features for recognizing previous and future repetitions. The proposed network is designed to extract the context-aware features from two consecutive repetitions, and a anchor-based backend is tailored for detecting double-error or half-error. The proposed temporal repetition counting framework is evaluated and compared with state-of-the-art methods and achieves better results in the existing benchmarks as well as our newly proposed dataset.

Acknowledgement

References

[1] A Branzan Albu, Robert Bergevin, and Sébastien Quirion. Generic temporal segmentation of cyclic human motion. *Pattern Recognition*, 41(1):6–21, 2008.

[2] Ousman Azy and Narendra Ahuja. Segmentation of periodically moving objects. In *ICPR*, pages 1–4, 2008.

[3] Alexia Briassouli and Narendra Ahuja. Extraction and analysis of multiple periodic motions in video sequences. *IEEE TPAMI*, 29(7):1244–1261, 2007.

[4] Gertjan J Burghouts and J-M Geusebroek. Quasi-periodic spatiotemporal filtering. *IEEE TIP*, 15(6):1572–1582, 2006.

[5] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015.

[6] Dmitry Chetverikov and Sándor Fazekas. On motion periodicity of dynamic textures. In *BMVC*, pages 167–176, 2006.

[7] Ross Cutler and Larry S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE TPAMI*, 22(8):781–796, 2000.

[8] James Davis, Aaron Bobick, and Whitman Richards. Categorical representation and recognition of oscillatory motion patterns. In *CVPR*, pages 628–635, 2000.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[10] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, pages 6546–6555, 2018.

[11] Shiyao Huang, Xianghua Ying, Jiangpeng Rong, Zeyu Shang, and Hongbin Zha. Camera calibration from periodic motion of a pedestrian. In *CVPR*, pages 3025–3033, 2016.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[13] Ivan Laptev, Serge J Belongie, Patrick Perez, and Josh Wills. Periodic motion detection and segmentation via approximate sequence alignment. In *ICCV*, pages 816–823, 2005.

[14] Ofir Levy and Lior Wolf. Live repetition counting. In *ICCV*, pages 3020–3028, 2015.

[15] Xiu Li, Hongdong Li, Hanbyul Joo, Yebin Liu, and Yaser Sheikh. Structure from recurrent motion: From rigidity to recurrency. In *CVPR*, pages 3032–3040, 2018.

[16] Xiaoxiao Li, Vivek Singh, Yifan Wu, Klaus Kirchberg, James Duncan, and Ankur Kapoor. Repetitive motion estimation network: Recover cardiac and respiratory signal from thoracic imaging. *arXiv preprint arXiv:1811.03343*, 2018.

[17] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*, pages 3–19, 2018.

[18] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei. Gaussian temporal awareness networks for action localization. In *CVPR*, June 2019.

[19] ChunMei Lu and Nicola J Ferrier. Repetitive motion analysis: Segmentation and event classification. *IEEE TPAMI*, 26(2):258–263, 2004.

[20] Costas Panagiotakis, Giorgos Karvounas, and Antonis Argyros. Unsupervised detection of periodic segments in videos. In *ICIP*, pages 923–927, 2018.

[21] Erik Pogalin, Arnold WM Smeulders, and Andrew HC Thean. Visual quasi-periodicity. In *CVPR*, pages 1–8, 2008.

[22] Yang Ran, Isaac Weiss, Qinfen Zheng, and Larry S Davis. Pedestrian detection via periodic motion analysis. *IJCV*, 71(2):143–160, 2007.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015.

[24] Evan Ribnick and Nikolaos Papanikolopoulos. 3d reconstruction of periodic motion from a single view. *IJCV*, 90(1):28–44, 2010.

[25] Tom FH Runia, Cees GM Snoek, and Arnold WM Smeulders. Real-world repetition estimation by div, grad and curl. In *CVPR*, pages 9009–9017, 2018.

[26] Tom FH Runia, Cees GM Snoek, and Arnold WM Smeulders. Repetition estimation. *IJCV*, 127(9):1361–1383, 2019.

[27] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, pages 1049–1058, 2016.

[28] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[29] Ashwin Thangali and Stan Sclaroff. Periodic motion detection and estimation via space-time sampling. In *WACV*, pages 176–182, 2005.

[30] Christopher J Tralie and Jose A Perea. (quasi) periodicity quantification in video data, using topology. *SIAM Journal on Imaging Sciences*, 11(2):1049–1077, 2018.

[31] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.