Movement Prediction of Three Bouncing Balls

Abstract

Recurrent neural network(RNN) has been widely used in the processing of sequential data, including tasks like video prediction and natural language processing. Specifically, ConvLSTM model[1] is powerful in learning image sequential information. However, training such models requires a high computational cost in terms of time and space. To address this challenge, we extend the classic convolutional neural network(CNN) with two additional layers called "tendency layer", which captures the differences between two consecutive image frames, and "attention layer", which learns to focus on the most relevant input frame. We evaluate our methods on two datasets, three bouncing balls and Weizmann human action[29], observe that our models generate comparative results as RNN model like ConvLSTM but with substantial improvement in training efficiency.

1 Introduction

The prediction task has recently been well studied by using RNN models. For instance, ConvLSTM, an LSTM-based model that has the ability to learn the spatial and temporal features simultaneously, is a natural fit in such a task. However, due to the underlying structure of LSTM, it usually requires a huge amount of time to train the model and is difficult to parallelize this process. On the other hand, the convolutional neural network can capture visual imagery with spatial information and have high training efficiency. Is it possible for CNN to represent the time-series data and tackle the task of image sequence prediction?

The main difficulty for CNN to learn the temporal data is that traditional CNN models only focus on interpreting a single image, instead of finding potential relationships between multiple inputs. Another problem is that even the model is able to learn the sequential information, we need to know which part of previous information is useful. For example, in the prediction of ball movement, a collision will change where the ball is moving to. Such important events have a great impact on the future, and our model should be able to distinguish those frames when those events happen.

In order to overcome these obstacles, we add two new layers to the traditional CNN models. The advantages of our design are that our model can utilize the difference between frames and assign importance weights on different parts of inputs, which to some extent capture the temporal information. Besides, we also improved the basic structure of ConvLSTM to fit it on image sequence prediction task both at input and output for performance comparison with our proposed new CNN model. We evaluated our method on the bouncing ball and a real-life human action dataset. The result demonstrates that our proposed CNN model obtains a comparable result to the ConvLSTM model while we also see a substantial improvement on the training efficiency.

2 Related Work

2.1 Video frame prediction

In the context of video frame prediction task, it has been shown that it is possible to use high-level embeddings to anticipate future actions up to one second before they begin [1]. Predicting the future event by retrieving similar videos and transferring this information, is proposed in [2]. In [3] a hierarchical representation is used for predicting future actions. Predicting a future activity based on analyzing object trajectories is proposed in [4]. Anticipating future movement in the spatial domain, as close as possible to the real movement, has also been previously considered. Here, the methods start from an input image at the current timestamp and predict motion optical flow or motion trajectories at the next frame of a video. In [5] images are aligned to their nearest neighbor in a database and the motion prediction is obtained by transferring the motion from the nearest neighbor to the input image. In [6], structured random forests are used to predict optical flow vectors at the next time stamp. In [7], the use of LSTM (Long Short-Term Memory Networks) is advised towards predicting Eulerian future motion. A custom deep convolutional neural network is proposed in [8] towards future optical flow prediction.

2.2 Sequential model

Recent advances in deep learning, especially recurrent neural network (RNN) and long short-term memory (LSTM) models [10, 11, 12, 13, 14, 15, 16, 17], provide some useful insights on how to solve the sequence prediction problem. The pioneering LSTM encoder-decoder framework proposed in [14] provides a general framework for sequence-to-sequence learning problems by training temporally concatenated LSTMs, one for the input sequence and another for the output sequence. In [16], it is shown that prediction of the next video frame and interpolation of intermediate frames can be done by building an RNN based language model on the visual words obtained by quantizing the image patches. They propose a recurrent convolutional neural network to model the spatial relationships but the model only predicts one frame ahead and the size of the convolutional kernel used for state-to-state transition is restricted to 1. Their work is followed up later in [17] which points out the importance of multi-step prediction in learning useful representations. They build an LSTM encoder-decoder predictor model which reconstructs the input sequence and predicts the future sequence simultaneously. Combining the most successful neural network paradigms for image data (CNN) and sequence modeling (LSTMs) to exploit spatiotemporal relations has led to fruitful applications of the above ideas to a variety of tasks. [18] propose a LSTM with convolutional(ConvLstm) instead of linear interactions and demonstrate good performance on predicting precipitation based on radar echo images.

2.3 Convolutional neural network

Renaissance of deep neural network remarkably accelerates the progress in image-related tasks. Convolutional Neural Networks (CNN) [19] can learn powerful features from large-scale image datasets, which greatly alleviates the difficulty of designing hand-crafted features. Extensive experiments have demonstrated that CNN can achieve superior performance in various image and video classification tasks, e.g. ImageNet object classification [20], face recognition [21]. These successes inspire researchers to extend CNN for video classification tasks [22, 23, 24]. Karpathy et al. [22] proposed several convolutional neural network (CNN) architectures based on stacked RGB images for video classification. They designed several fusion strategies for RGB flows to utilize temporal information in stacked RGB frames. [24] modeled temporal information by designing a 3D filter to directly learn feature from videos. Simonyan and Zisserman [23] proposed the two-stream architecture which exploits two CNN to model RGB and optical flow respectively. Unlike these video classification task, we want to illustrate if there are any possibilities to utilize CNN structure to represent temporal information in video frames and predict next frames based on some earlier frames.

3 Methodology

3.1 Model Structure

We've tried three different models, one of them is based on ConvLSTM, and the other two of them are based on simple CNN structure (LeNet). The three model structures are described as follows:

(1) Time-Series CNN

We first build a Simple CNN Encoder-decoder model for high-dimensional sequential inputs and outputs. Then to help CNN better learn from the time-series data, 2 hand-designed layers, Movement Tendency Layer and Attention Layer, are combined into the encoder. CNN has great performance on extracting features from pictures. Here we fine-tuned LeNet-5[27] as our encoder. As its known, LeNet-5 is designed for the classification task, so the output is only 10 nodes. Therefore, we add fully connected layers as our decoders and the number of output nodes is adjustable based on the number of output frames wanted. Movement Tendency Layer is computed by subtracting former layers from latter layers. In the Movement Tendency Layer, the non-trivial background is removed, and the remainder are significant movement trend of the object. The movement tendency information is feed into CNN and combined into our encoder. Inspired by Transformer [28] model, we design an attention mechanism for the input frames by adding learnable weights on each input frames. Those weights are implemented by depth-wise convolution filters, which also extract features at the same time and save the number of parameters.

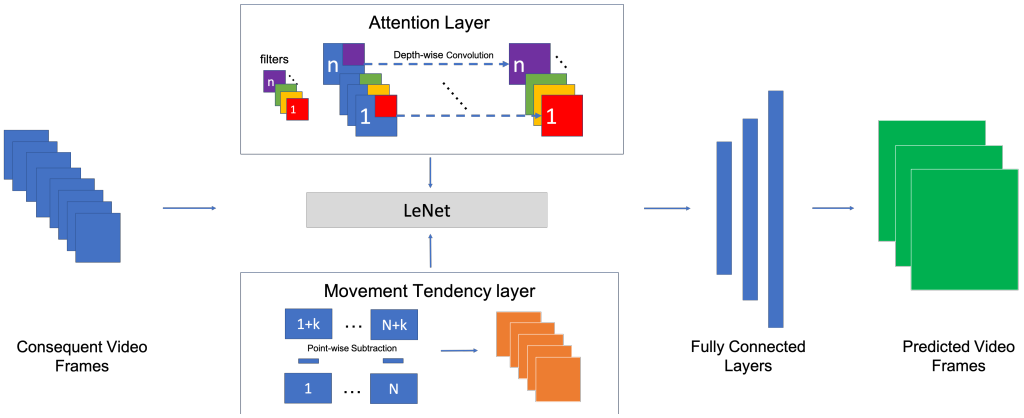


Figure 1: Time-Series CNN model structure

(2) Difference CNN Encoder-Decoder

For the time-series CNN encoder-decoder model, we noticed that it was not easily adapted to this new dataset. Instead of inheriting the same structure of that model, we decided to propose a new design specifically for this dataset, which is based on the same idea as before but has a different architecture, named as Difference CNN Encoder-Decoder Model(Figure 1). Unlike our simple CNN encoder-decoder model or works of Long et al. [26] and Haitam Ben Yahia[25], we do not feed a sequence of frames as channel into the encoder, instead, we compute the difference of each input frame and its next frame, and use those difference images as inputs. We also have a detector structure, where we feed in the last frame of the input sequence. The detector part is a LeNet-based CNN and its output is also an encoding vector but has a much larger size, for example, 600*1 for the detector and 200*1 for the encoder.

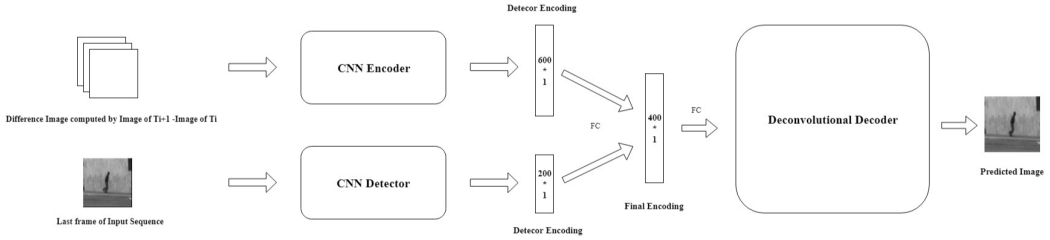


Figure 2: Difference CNN Encoder-Decoder model structure

(3) Stacked Conv-Deconv-LSTM

In the Stacked Conv-Deconv-LSTM, we keep the basic LSTM cell and the convolutional encoders for input images at different time scale. In addition, we refer to the idea of "FCN"[29], a famous neural network for image segmentation task and add the deconvolutional layer to proceed the output of LSTM cell. These deconvolutional layers can make the final output same size with the original input by setting appropriate filter parameters, which make it easier to compare the difference between ground truth and our prediction. Besides, we also refer to the idea of "Stacked LSTM"[30] to build a deep LSTM by stacking multiple recurrent hidden states on top of each other. This approach potentially allows the hidden state at each level to operate at different timescale.

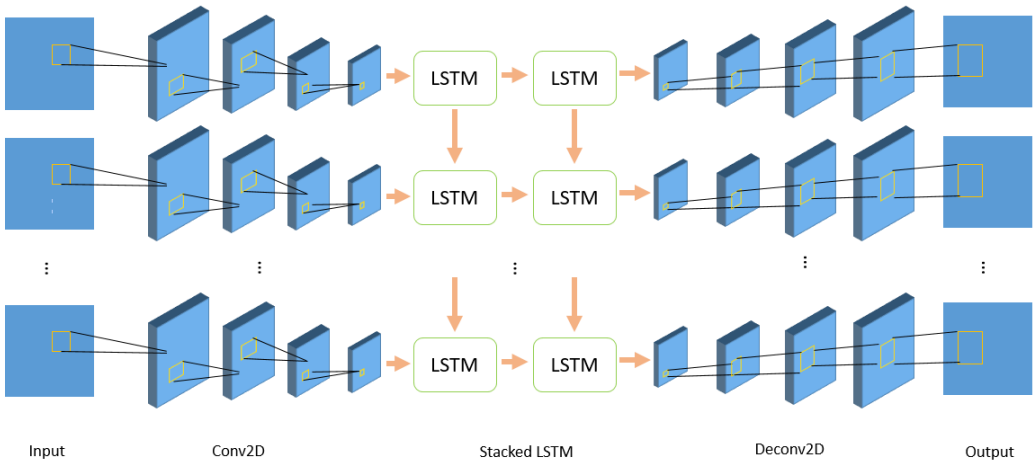


Figure 3: Stacked Conv-Deconv-LSTM

3.2 Dataset

We test our models on two different datasets. The first one is the bouncing ball dataset, which is a common test set for models that generate high dimensional sequences. It consists of simulations of three balls bouncing in a box. We followed the standard procedure to create 15000 training samples and 5000 testing samples. Our networks were trained with 5 frames as input, and output 1 and 3 predicted frames respectively. Training with MSE was very effective for this dataset. In order to test the universality of our models, we experimented our models on the Weizmann human action dataset. Each video has a noisy but stable background (the wall and the window) with moving objects (the human), therefore it is desirable to serve as a probe for testing the ability to capture the motions. Each sample is composed of 5 consecutive frames of one video and the following 1, 3, 5 frames used as our prediction targets. In total, there are 3961 samples. We divide the dataset into two parts, with 3600 samples as training data and the rest as testing data.

3.3 Metrics

(1) Mean Square Error (MSE) Mean Square Error describes the global similarity of the prediction and ground truth. We also adopt it as the loss function for its high efficiency on training. Lower is better.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$$

(2) Peak Signal-to-Noise Ratio (PSNR) Peak signal-to-noise ratio(PSNR) is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation[6]. The PSNR value can be easily computed given MSE:

$$PSNR = 10 * \log_{10}(\frac{R^2}{MSE})$$

(3) Structural Similarity Index (SSIM) SSIM is another way to measure the similarity of two pictures. Higher is better. SSIM is designed to improve on traditional methods such as PSNR and MSE. It can be considered as a good supplementary evaluation metric for measuring the quality of predictions.

4 Experiment

4.1 Evaluation on bouncing ball dataset

We use 15000 samples to train each model and 5000 samples to test. For every model, we input 5 sequential frames, and let the model predict the following 1 or 3 frames.

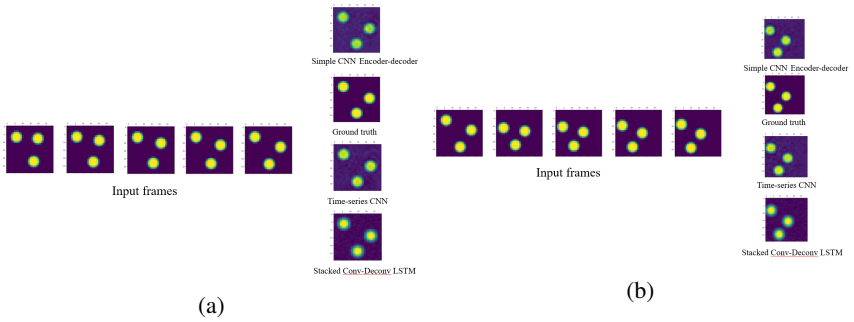


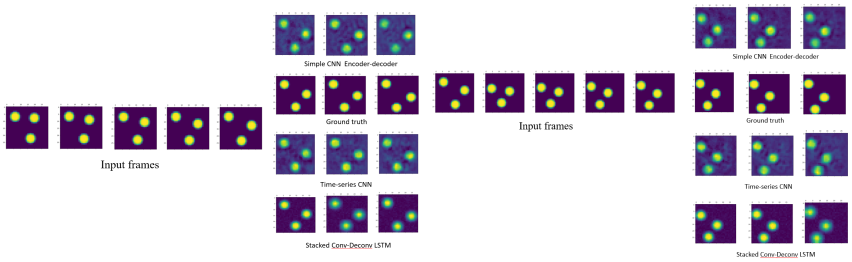
Figure 4: One frame prediction on two different inputs

Method/Metrics	MSE	PSNR	SSIM
Simple CNN	157.55	28.08	0.42
Time-series CNN	154.59	29.49	0.43
Stacked Conv-LSTM	169.27	27.60	0.38

Table 1: Metrics for 1 frame prediction

Both Simple CNN Encoder-Decoder and Time-series CNN performs well on 1-frame prediction. The MSE, PSNR and SSIM are almost the same for these two models.

According to the pictures, Time-Series CNN has better prediction with respect to the balls, while Simple CNN Encoder-Decoder predicts clearer background.



(a) (b)

Figure 5: Three frames prediction on two different inputs

Method/Metrics	MSE	PSNR	SSIM
Simple CNN	866.58	19.61	0.33
Time-series CNN	842.16	21.39	0.37
Stacked Conv-LSTM	298.24	35.55	0.41

Table 2: Metrics for 3 frame prediction

Time-series CNN performs much better than Simple CNN Encoder-Decoder when predicting 3 frames. Stacked Conv-LSTM becomes significantly powerful when predicting 3 frames.

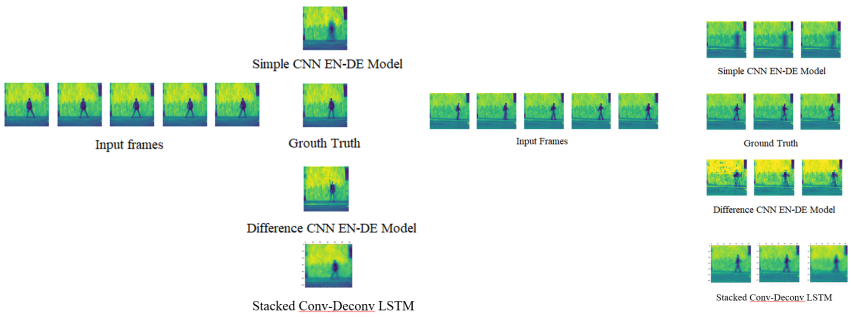
For computational cost, Simple CNN Encoder-decoder Model takes about 20-30 minutes for training to be finished (200 epochs). Time-series CNN takes about 40-50 minutes for training to be finished (200 epochs). However, Stacked Conv-Deconv LSTM usually takes more than 2 hours finishing the training process with a good performance (200 epochs). The CNN-based model has great advantages on training efficiency over LSTM-based model.

4.2 Evaluation on human action dataset

We first experimented our models to predict one frame given 5 input frames. Table 1 shows that a comparison of the prediction quality of three measurement metrics. CNN CN-DE denotes our Simple CNN encoder-decoder model, Diff CNN EN-DE is the Difference CNN Encoder-Decoder model that we designed for this task. Baseline takes the last frame of inputs as the prediction. Our Simple Encoder-Decoder model achieves the second low MSE score. Our Difference encoder-decoder performs badly in all three metrics. Conv-LSTM based model performs badly maybe due to the hyperparameters are not finetuned in this task.

Method/Metrics	MSE	PSNR	SSIM
Simple CNN	117.3	27.44	0.852
Diff CNN	200.1	25.12	0.786
Stacked Conv-LSTM	325.63	28.24	0.42
Baseline:Last Frame	78.4	29.19	0.951

Table 3: Metrics for 1 frame prediction



(a) One frame prediction

(b) Three frame prediction

Figure 6: Frame prediction on human action dataset

Some visualization results are shown in figure 6. As you can see from the (a), the result given by our simple CNN encoder-decoder model seems like an aggregation over previous images, while our difference CNN encoder-decoder generates a more reasonable result where the outline of the man is clear and its position is right to the inputs(the running direction). However, since our difference CNN encoder-decoder model has a bad background prediction, it ends up with a high MSE.

We further experimented our models to predict 3 frames given 5 input frames. In order to predict 3 frames, we use the last prediction result as input and predict the next one. Table 4 shows a comparison of the quality of predicting 3 frames between our two methods. Our simple CNN encoder-decoder model outperforms our Difference CNN model for predicting both 3 frames, in terms of all three metrics. However, the same problem persists. In Figure 6 (b), our simple CNN encoder-decoder seems like mixing all input frames together and compute the average. It neither demonstrates the motions nor outlines the human shape. Our Difference CNN encoder-decoder model generates a better shape of the human, and demonstrates the tendency to movement, but the background is not realistic enough. Since the background weighs highly in the computation of all three metrics, such good property cannot be reflected in numbers.

Method/Metrics	MSE	PSNR	SSIM
Simple CNN	132.3	26.91	0.857
Diff CNN	197.3	25.18	0.761
Stacked Conv-LSTM	377.13	29.92	0.392

Table 4: Metrics for 3 frame prediction

5 Conclusion

From our experiment with different models we proposed, we found that time-series CNN has generally better result than Simple CNN Encoder-Decoder on sequence prediction. Both CNN-based model performs well on 1-frame prediction. But Time-series CNN become more powerful when predicting multi-frames. Stacked Conv-Deconv LSTM has the best result most of time and is more natural to generate long sequences. Tradeoff has to be made between time and accuracy. Time-series CNN has much faster training speed while LSTM based model has more accurate prediction result.

Contribution

References

[1] Xingjian, S. H. I., et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." Advances in neural information processing systems. 2015.

[2] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. CoRR, 2015.

[3] J. Yuen and A. Torralba. A data-driven approach for event prediction. In ECCV, pages 707720. Springer, 2010.

[4] T. Lan, T. C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In ECCV, pages 689704. Springer, 2014.

[5] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In ECCV, pages 201214. Springer, 2012

[6] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. PAMI, 33(5):978994, 2011.

[7] S. L. Pintea, J. C. van Gemert, and A. W. M. Smeulders. Deja vu: Motion prediction in static images. In ECCV, pages 172187. Springer, 2014.

[8] S. L. Pintea and J. C. van Gemert. Making a case for learning motion representations with phase. In ECCVw, 2016

[9] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In ICCV, pages 24432451, 2015.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):17351780, 1997.

[11] A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[12] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, pages 1724 1734, 2014.

[13] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, 2015.

[14] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In NIPS, pages 31043112, 2014.

[15] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In CVPR 2015.

[16] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604, 2014.

[17] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In ICML, 2015.

[18] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In Advances in Neural Information Processing Systems, pages 56225632, 2017.

[19] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS12, pages 10971105, 2012.

[21] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In CVPR14, pages 17011708, 2014.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR14, pages 17251732, 2014.

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In NIPS14, pages 568576, 2014.

[24] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. CoRR, abs/1412.0767, 2014.

[25] Yahia, Haitam Ben, K. Intelligentie, and M. Reisser. Frame interpolation using convolutional neural networks on 2d animation. Diss. Ph. D. Thesis, MA thesis. University of Amsterdam. Google Scholar, 2016.

[26] Long, Gucan, et al. "Learning image matching by simply watching video." European Conference on Computer Vision. Springer, Cham, 2016.

[27]. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[28]. Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.

[29] Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as Space-Time Shapes. ICCV (2005)