Seeing the Arrow of Time

CV Project

Team :- Beam Search

Rahul Manoj 201501151
Tarun Gavara 201501147
Pandramish Vinay 201425130

Introduction

- The Arrow of Time is a concept developed in 1972 by Arthur Eddington. This paper deals with the psychological/perceptual arrow of time.
- We are interested in finding the Arrow of Time i.e whether a particular video is playing forwards or backwards. We use the underlying physics to find it, but not object level visual cues.
- Time’s Arrow is a special case of causal inference related to the field of Computer Vision.
- Finding Time’s Arrow is useful in forensics.

Data Set Used For This Project

- The paper mentions a 155 forward + 25 backward video dataset obtained from YouTube.
- These videos are HD (thus allowing to subsample) and are free of CGI, animation, special effects, excessive camera motion, motion blur and bad lighting as the underlying physics describing these may not match exactly with the real-world physics.
- There is also a 13 video collection of tennis balls being rolled.

Baseline Performance

- Spatial-temporal Orientated Energy (SOE) is an off-the-shelf method which can be employed to describe video motion. Each video is split into 2 × 2 spatial sub-regions, and the SOE responses for each subregion are concatenated to form a final feature.
- Using these SOE features in a linear SVM classifier, the baseline performance on the three splits of the YouTube dataset described above are 57%, 48% and 60% respectively.
- The relatively poor performance can be attributed to the difficulty in generalizing motion over different sub regions, and the fact that these features were designed to characterize regular motion textures, rather than particular one-off dynamic events.

Methods Proposed

There were three methods proposed to find out the arrow of time :-

1. Statistical Flow Method.
2. Motion Causation Method.
3. AR Method.

Statistical Flow Method

- Look at the occurrences of particular motion patterns across a time-sequence.
- First register the frames of a video in order to compensate for hand-shake and intentional camera motion. Assume that any residual motion is due to objects moving in the scene.
- Compute SIFT using motion gradients resulting in a descriptor which represents local histograms of image motion.
- These object-motion descriptors are then vector-quantized to form a discrete set of flow words, and a bag-of- flow-word descriptor representing the entire video sequence is computed.
- With sufficient forward and reverse-time examples, a classifier can be trained to discriminate between the two classes.

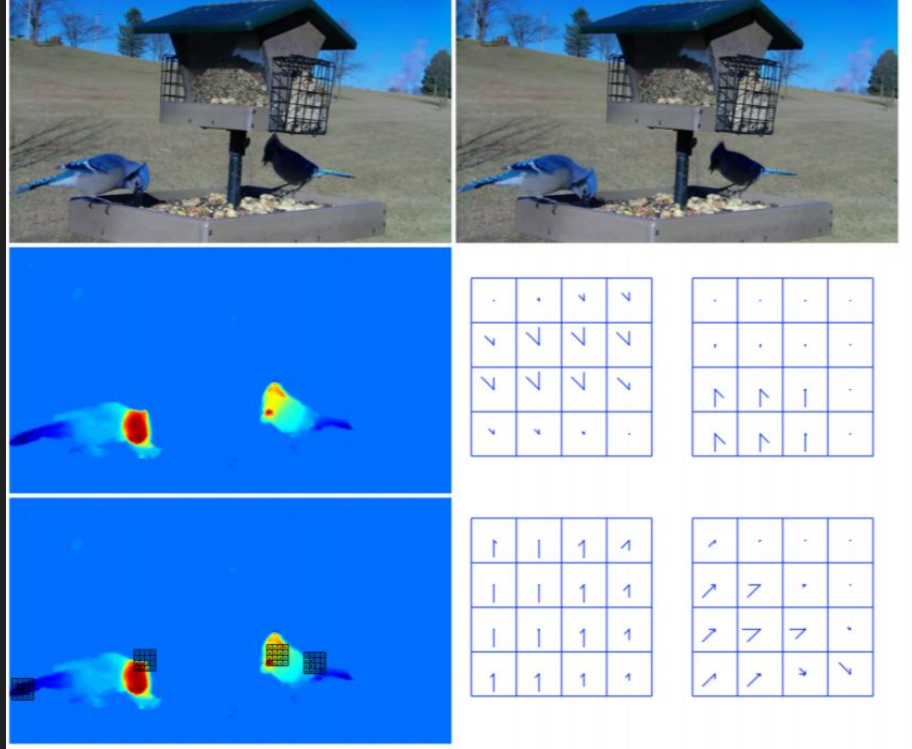

Figure 2. **Construction of Flow-words features**. Top: pair of frames at times $t-1$ and $t+1$, warped into the coordinate frame of the intervening image. Left: vertical component of optical flow between this pair of frames; lower copy shows the same with the small SIFT-like descriptor grids overlaid. Right: expanded view of the SIFT-like descriptors shown left. Not shown: horizontal components of optical flow which are also required in constructing the descriptors.

Motion Causation Method

- We consider the fact that motion causes other motions. We could say that the entropy of motion increases for a forward playing action. Like when a cue hits a pool ball, every ball starts moving.
- The backwards playing video would be like a bunch of balls falling into a shape of a triangle and stopping perfectly along with the cue going back and that would not be naturally possible.
- We look at the regions in the video from frame to frame where differences are appearing, Smoothed regions of motion in one frame are compared to similar regions in the previous frame.
- We expect more occurrences of one region splitting in two than of two regions joining to become one, in the forwards-time direction.

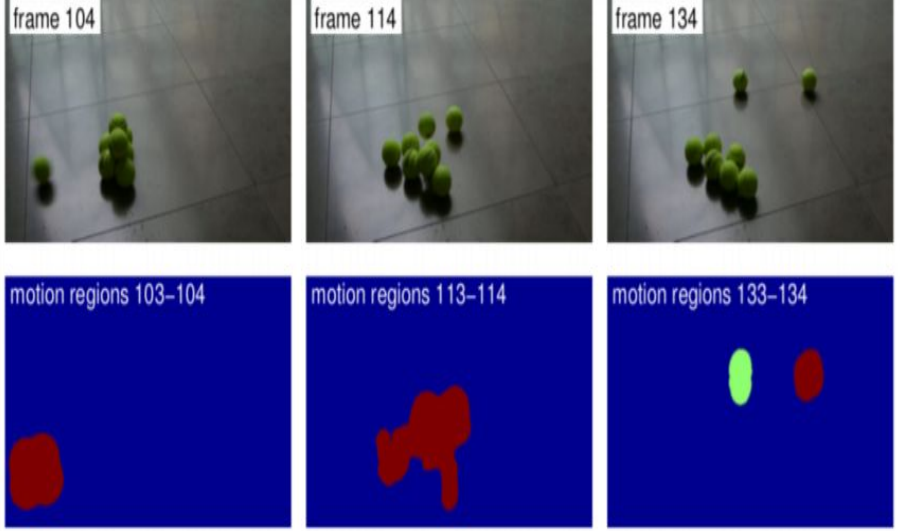

Figure 6. Three frames from one of the Tennis-ball dataset sequences, in which a ball is rolled into a stack of static balls. Bottom row: regions of motion, identified using only the frames at t and $t-1$. Notice that the two rolling balls are identified as separate regions of motion, and colored separately in the bottom right-most plot. The fact that one rolling ball (first frame) causes two balls to end up rolling (last frame) is what the motion-causation method aims to detect and use.

AR Method

- For non-Gaussian additive noise and dynamics obeying a linear ARMA (autoregressive moving average) model, the noise added at some point of time is independent of the past values of the time series, but not of the future values. This allows us to determine the direction of time by independence testing.
- The assumption that some image motions will be modelled as AR models with additive non-Gaussian noise leads to a simple algorithm for measuring the direction of time in a video: track the velocities of moving points, fit those velocities with an AR model and perform an independence test between the velocities and model residuals (errors).

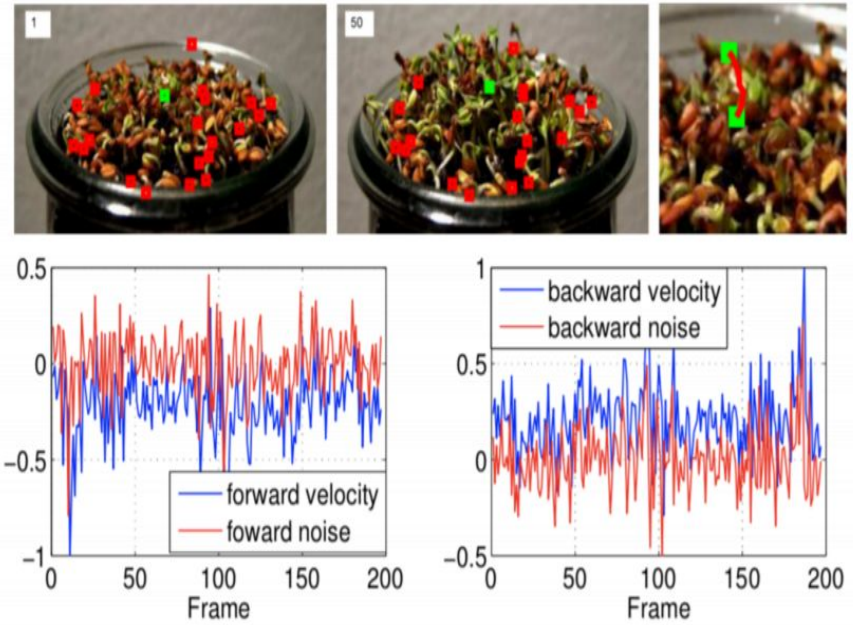

Figure 7. Overview of the AR method. Top: tracked points from a sequence, and an example track. Bottom: Forward-time (left) and backward-time (right) vertical trajectory components, and the corresponding model residuals. Trajectories should be independent from model residuals (noise) in the forward-time direction only. For the example track shown, p-values for the forward and backward directions are 0.5237 and 0.0159 respectively, indicating that forwards time is more likely.

Implementation of Statistical Flow

```
def kfold_run(path):

# loop through the given dataset

        # build_dict_of_flow_words(path)  -   0

        # for each video in train_dataset

                # get_A_hist_of_video, store it in hist_A, lly generate label_A     -    1
                # get_B_hist_of_video, store it in hist_B, lly generate label_A
                # get_C_hist_of_video, store it in hist_C, lly generate label_A
                # get_D_hist_of_video, store it in hist_D, lly generate label_A

        # save the dataset
        # perform pca on each of the hists individually        -      2

        # train the dataset on MLP, to give A,B,C,D values given an input of all hist combined together    -     1
        # repeat the above steps with the test data except, instead of training on the model predict

    # reset the weights, vocab
```

Implementation

The videos are split according to the input file's fname[0] value which is either 'F' for forward and 'B' for backward. We randomise the data to maintain the soundness. After this, we build the dict for forward and backward videos separately using the build_dict() function. We finally make sure that the dict_of_flow_words, which consists of all the vertical and horizontal optical flow values, is greater than the threshold.

The threshold here helps to remove the motion values due to the static difference between the frames rather than due to the dynamic motion changes that we require. Using K-Means we learn the dictionary of words and then the descriptors from the motion images are assigned to the closest word. Then we represent the video sequence as a normalized histogram of visual flow words across the entire time-span of the video. For each video the A, B, C and D versions we extract the descriptor histograms.

Implementation (cont.)

This information is stored in four different arrays. This code is present in the createNNinputsPCA folder which also extracts only for the most important 60 components. And finally, we run an SVM across all of this processed information for classification. For a valid classification A + B - C - D should output positive for forward motion in video and negative output for backward motion.
We also run a nearest neighbour training algorithm using keras for optimisation.

Results

In conclusion, we have used softmax over a neighbourhood instead of sift-like descriptors. We then used PCA to compress the histograms of each video. Then we used a 3 layer Multi Layer Perceptron along with SVM to compare the performance.

We used the dataset given in the paper: train dataset of 70 videos and a test dataset of 60 videos.
The accuracy published in the paper was [90% on train:test => 120:60]
The MLP accuracy was [99.29% on train:test => 70:60].


Thank
you