

Seeing the Arrow of Time

Lyndsey C. Pickup¹ Zheng Pan² Donglai Wei³ YiChang Shih³ Changshui Zhang²
Andrew Zisserman¹ Bernhard Schölkopf⁴ William T. Freeman³

¹Visual Geometry Group, Department of Engineering Science, University of Oxford.

²Department of Automation, Tsinghua University State Key Lab of Intelligent Technologies and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList).

³Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

⁴Max Planck Institute for Intelligent Systems, Tübingen.

Abstract

We explore whether we can observe Time’s Arrow in a temporal sequence—is it possible to tell whether a video is running forwards or backwards? We investigate this somewhat philosophical question using computer vision and machine learning techniques.

We explore three methods by which we might detect Time’s Arrow in video sequences, based on distinct ways in which motion in video sequences might be asymmetric in time. We demonstrate good video forwards/backwards classification results on a selection of YouTube video clips, and on natively-captured sequences (with no temporally-dependent video compression), and examine what motions the models have learned that help discriminate forwards from backwards time.

1. Introduction

How much of what we see on a daily basis could be time-reversed without us noticing that something is amiss? Videos of certain motions can be time-reversed without looking wrong, for instance the opening and closing of automatic doors, flexing of the digits of a human hand, and elastic bouncing or harmonic motions for which the damping is small enough as to be undetectable in a short video. The same is not true, however, of a cow eating grass, or a dog shaking water from its coat. In these two scenarios, we don’t automatically accept that a complicated system (chewed food, or the spatter pattern of water drops) will re-compose itself into a simpler one.

There has been much study of how the underlying physics, and the second law of thermodynamics, create a direction of time [14, 16]. While physics addresses whether the world is the same forwards and backwards, our goal will be to assay whether and how the direction of time manifests

itself *visually*. This is a fundamental property of the visual world that we should understand. Thus we ask the question: *can we see the Arrow of Time? – can we distinguish a video playing forward from one playing backward?*

Here, we seek to use low-level visual information – closer to the underlying physics – to see Time’s Arrow, not object-level visual cues. We are not interested in memorizing that cars tend to drive forward, but rather in studying the common temporal structure of images. We expect that such regularities will make the problem amenable to a learning-based approach. Some sequences will be difficult or impossible; others may be straightforward. We want to know: how strong is the signal indicating Time’s Arrow? Where do we see it, and when is it reliable? We study these questions using several different learning-based approaches.

Asking such fundamental questions about the statistical *structure* of images has helped computer vision researchers to formulate priors and to develop algorithms exploiting such priors. For example, asking, “are images invariant over scale?” led researchers to develop the now widely-used multi-resolution spatial pyramid architectures [3]. Human vision researchers have learned properties of human visual processing by asking “can we recognize faces upside down?” [19], or about tonescale processing by asking, “can we recognize faces if the tonescale is inverted?” [9]. Both symmetries and asymmetries of priors provide useful information for visual inference. The prior assumption that light comes from above is asymmetric and helps us disambiguate convex and concave shapes [15]. Spatial translation invariance, on the other hand, allows us to train object detection or recognition systems without requiring training data at all possible spatial locations.

Learning and representing such statistical structure has provided the prior probabilities needed for a variety of computer vision tasks for processing or synthesizing images and image sequences, including, for example: noise removal

[17], inpainting over spatial or temporal gaps [1, 24], image enhancement [25], and texture synthesis [13].

In contrast to quite extensive research on the statistical properties of the *spatial* structure in images [6], there has been far less investigation of *temporal* structure (one of the exceptions to this rule includes [7]). However, knowledge of the temporal structure of images/videos can be used in a variety of tasks, including video enhancement or analysis, filling-in missing data, and the prediction of what will be seen next, an important feedback signal for robotics applications. Thus, we can similarly ask: *are video temporal statistics symmetric in time?*

We expect that the answer to this question can guide us as we develop priors and subsequent algorithms. For example, an understanding of the precise asymmetries of temporal visual priors may have implications for methods for video denoising or video decompression or optical flow estimation or ordering of images [4]. In addition, causal inference, of which seeing Time’s Arrow is a special case, has been connected to machine learning topics relevant to computer vision, such as semi-supervised learning, transfer learning and covariate shift adaptation [18]. Our question relates to this issue of causality, but seen through the lens of vision.

In the sequel, we probe how Time’s Arrow can be determined from video sequences in three distinct ways: first, section 3, by using features that represent local motion patterns and learning from these whether there is an asymmetry in temporal behaviour that can be used to determine the video direction; second, section 4, by explicitly representing evidence of causality; and third, section 5, by using a simple auto-regressive model to distinguish whether a cause at a time point influences future events or not (since it can’t influence the past).

2. Dataset and baseline method

In this section we describe the two video datasets and a baseline method for measuring Time’s Arrow.

2.1. YouTube dataset

We have assembled a set of videos that act as a common test bed for the methods investigated in this paper. Our criteria when selecting videos is that the information on Time’s Arrow should not be ‘semantic’ but should be available principally from the ‘physics’. Thus we do not include strong semantic cues for example from the coupling of the asymmetries of animals and vehicles with their directions of motion (e.g. eyes at the front of the head tending to move forwards); or from the order of sequences of actions (such as traffic lights or clock hands). What remains is the ‘physics’ – gravity, friction, entropy, etc [21].

This dataset consists of 180 video clips from YouTube. This was obtained manually using more than 50 keywords, aimed at retrieving a diverse set of videos from which we

might learn low-level motion-type cues that indicate the direction of time. Keywords included “dance”, “steam train”, and “demolition”, amongst other terms. The dataset is available at <http://www.robots.ox.ac.uk/data/arrow/>.

Selection criteria: Video clips were selected to be 6–10 seconds long, giving at least 100 frames on which to run the computations. This meant having to discard clips from many professionally-produced videos where each shot’s duration was too short, since each clip was required to be a single shot. We restricted the selections to HD videos, allowing us to subsample extensively to avoid block artifacts and minimize the interaction with any given compression method.

Videos with poor focus or lighting were discarded, as were videos with excessive camera motion or motion blur due to hand shake. Videos with special effects and computer-generated graphics were avoided, since the underlying physics describing these may not match exactly with the real-world physics underpinning our exploration of Time’s Arrow. Similarly, split-screen views, cartoons and computer games were all discarded. Finally, videos were required to be in landscape format with correct aspect ratios and minimal interlacing artefacts. In a few cases, the dataset still contains frames in which small amounts of text or television channel badges have been overlaid, though we tried to minimize this effect as well.

Finally, the dataset contains a few clips from “backwards” videos on YouTube; these are videos where the authors or uploaders have performed the time-flip before submitting the video to YouTube, so any time-related artefacts that might have been introduced by YouTube’s subsequent compression will also be reversed in these cases relative to the forward portions of our data. In these cases, because relatively few clips meeting all our criteria exist, multiple shots of different scenes were taken from the same source video in a few cases. In all other cases, only one short clip from each YouTube video was used.

In total, there are 155 forward videos and 25 reverse videos in the dataset. Two frames from each of six example videos are shown in Figure 1. The first five are forwards-time examples (horse in water, car crash, mine blasting, baseball game, water spray), and the right-most pair of frames is from a backwards-time example where pillows are “un-thrown”.

Train/val/test split and evaluation procedure. For evaluating the methods described below, the dataset was divided into 60 testing videos and 120 training videos, in three different ways such that each video appeared as a testing video exactly once and training exactly twice. Where methods required it, the 120 videos of the training set were further subdivided into 70 training and 50 validation videos. The three



Figure 1. Six examples of videos from our 180-clip YouTube dataset, including one reverse-time example (right-most frames).

train/test splits are labeled A, B and C, and are the same for each of the methods we report. The backwards-to-forwards video ratios for the three test sets were 9:51, 8:52 and 8:52 respectively.

The evaluation measure for each method is the proportion of the testing videos on which the method could correctly predict the time-direction.

2.2. Tennis-ball dataset

We also filmed a small number of video clips using a camera which could record in a video codec which used only intra-frame, rather than inter-frame coding, meaning that there was no possibility of compression artefacts holding any time-direction information. This dataset comprises 13 HD videos of tennis balls being rolled along a floor and colliding with other rolling or static balls. An example of one of these sequences is used later in Figure 6.

2.3. Baseline method

We consider Spatial-temporal Orientated Energy (SOE) [5] as an off-the-shelf method which can be employed to describe video motion. SOE is a filter-based feature for video classification, and is sensitive to different spacetime textures. Our implementation is faithful to that laid out in [5], and comprises third-derivative-of-Gaussian filters in eight uniformly-distributed spatial directions, with three different temporal scales. As in the prior work, we split each video into 2×2 spatial sub-regions, and concatenated the SOE responses for each sub-region for 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.

3. Statistical flow method

Without semantic information, the most telling cues for the Arrow of Time in video data may well come from pat-

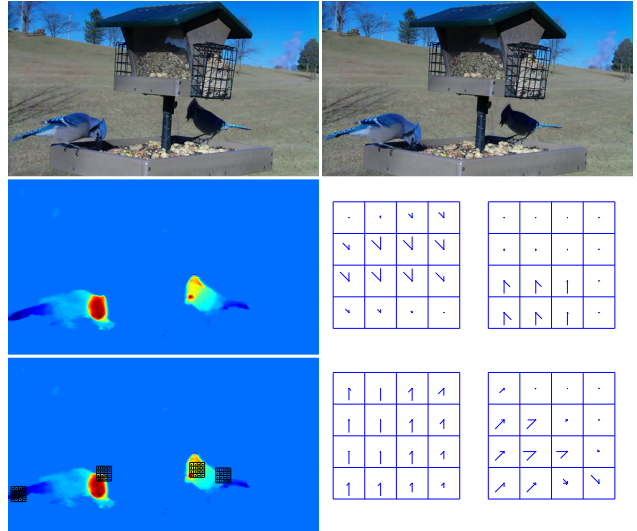


Figure 2. **Construction of Flow-words features.** Top: pair of frames at times $t - 1$ and $t + 1$, warped into the coordinate frame of the intervening image. Left: vertical component of optical flow between this pair of frames; lower copy shows the same with the small SIFT-like descriptor grids overlaid. 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.

terns of motion. So can simply looking at the occurrences of particular motion patterns across a time-sequence give us clues to Time’s Arrow?

There are many other subtle physical reasons why the forward and reverse time directions may look different on average. Friction and damping will tend to have characteristic effects on dynamic systems, and the reverse of these sorts of processes – objects taking in small amounts of heat energy from their surroundings in order to accelerate themselves – are far less likely to happen, so the way in which objects move is very relevant.

We propose a method we call *Flow-Words* in order to capture local regions of motion in a video so that we can examine which types of motion exhibit temporal asymmetries that might be useful for detecting the Arrow of Time.

Flow-words are based on a SIFT-like descriptor of mo-

tion occurring in small patches of a video. We first register the frames of a video in order to compensate for hand-shake and intentional camera motion (panning and zooming *etc.*), and then we assume that any residual motion is due to objects moving in the scene. Rather than computing a SIFT based on image edge gradients, motion gradients from an optical flow computation as substituted, giving a descriptor which represents local histograms of image motion. An example of the descriptors used to build these flow-words is shown in Figure 2.

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 thus computed. With sufficient forward and reverse-time examples, a classifier can be trained to discriminate between the two classes.

Implementation details. In a similar manner to [23], computation starts from dense trajectories. However, instead of only representing the histogram of optical flow over a region, as in the HOF features of [10], we also represent its spatial layout in a local patch in a manner similar to a local SIFT descriptor.

In detail, frames were downsized to 983 pixels wide in a bid to remove block-level artefacts and to provide a consistent image size for the rest of the pipeline. Images at $t - 1$ and $t + 1$ were registered to the image at time t , and an optical flow computation carried out [2]. This was repeated over the video sequence in a temporal sliding window, *i.e.* giving $T - 2$ optical flow outputs for a sequence of length T . These outputs take the form of motion images in the horizontal and vertical directions. A normal VLFeat dense SIFT descriptor [22] uses intensity gradient images in the x- and y-directions internally in its computations, so to describe the flows instead of the image structure, we simply operated on the vertical and horizontal motion images instead of the intensity gradients. This yielded motion-patch descriptors whose elements represented the magnitudes of motion in various directions passing through the t^{th} frame, on a small 4×4 grid (bin size 6 pixels) sampled once every 3 pixels in the horizontal and vertical directions.

We suppressed static trajectories, because we were only interested in dynamic behaviour for this work. This was achieved by setting the motion-gradient images to zero where the image difference was below a threshold, and not including motion-patch descriptors from such areas in further processing. This excised large areas, as can be seen in the constant background color in Figure 2.

Divergence operator for flow-words: In analyzing the results, it will be useful to quantify the divergence or other motion-field properties of a given flow-word descriptor. Using the divergence theorem, we make an approximation to the integral of the divergence over a descriptor by consid-

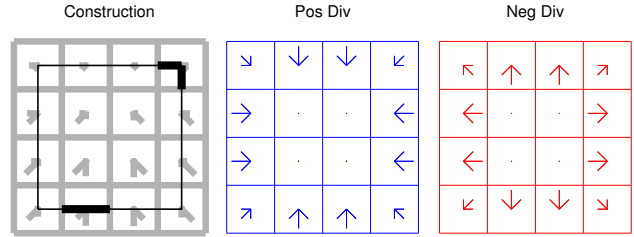


Figure 3. Left: construction of the divergence operator by summing contributions along straight-edge segments of a square superimposed on the flow-word grid. Centre & right: positive and negative elements of the div operator, respectively.

ering the net outward and inward motion fluxes across a square that joins the centers of the four corner cells on the descriptor grid, assuming that motion at every point within a grid cell is drawn independently from the histogram which is represented at the centre of that cell. The resulting divergence operator for flow-words is shown in the central and right parts of figure 3, with the negative and positive flux contributions separated for visualization.

3.1. Learning

A dictionary of 4000 words was learnt from a random subset of the training data ($O(10^7)$ examples) using K-means clustering, and each descriptor was assigned to its closest word. A video sequence was then described by a normalized histogram of visual flow-words across the entire time-span of the video. Performance was improved if the square roots of the descriptor values were taken prior to clustering, rather than the raw values themselves.

For each video in the training/validation set, we extracted four descriptor histograms: (*A*): the native direction of the video; (*B*): this video mirrored in the left-right direction; (*C*): the original video time-flipped; and (*D*): the time-flipped left-right-mirrored version. An SVM was trained using four histograms *A–D* extracted from each video of a training set, 280 (4×70) videos in total. Similarly, the 50 videos of the validation set generate 200 histograms *A–D* in total, and each is classified with the trained SVM. For a valid classification, *A* and *B* must have one sign, and *C* and *D* another. We combine the SVM scores as $A + B - C - D$, and this should give a positive score for forwards clips, a negative score for backwards clips, and some chance to self-correct if not all of the component scores are correct.

The *C* parameter for the SVM was chosen to be the value that maximized the classification accuracy on the validation set over all three train/test splits. Once this *C* parameter was fixed, the whole set of 120 training videos, including the previously-withheld validation set, was used to learn a final SVM for each train/test split.

Testing proceeds in a similar manner to that used for the validation set: the SVM scores for the four bag-of-flow-

word representations for each testing video are combined as $A + B - C - D$, and the sign of the score gives the video’s overall classification.

3.2. Experimental results

Training and testing were carried out according to the procedure set out in 3.1. The three trained models were used to classify the 60 videos in the three testing partitions, and achieved accuracies of 90%, 77% and 75% for splits A to C respectively.

Of the videos that were mis-classified, a few have lots of individuals moving separately in the shot: groups of people or animals, or a large number of small balls, which might not be well-suited for the scale of feature we are using. A further three were stop-motion films of seeds growing, which tend to have jerkier motions than most normal videos.

Tests on the Tennis-ball dataset were run in a leave-one-out manner, because of the small size of the dataset. The model was trained on 12 videos (and their various flips), and then tested on the withheld video, and all 13 possible arrangements were run. The correct time direction was found in 12/13 cases, *i.e.* over 90% accuracy.

3.3. Further analysis of Flow-words results

We can use the trained flow-words model to look at which flow-words were particularly characteristic of forwards or backwards motion. This was achieved by taking the SVM weights fixed using the training and set, and applying them to the descriptors for the test videos in their native playing direction. The 10 words (0.25% of the total vocabulary size) contributing most positively to the scores for forwards-direction clips, or most negatively for the backwards-direction clips, were recorded.

Figures 4 and 5 show flow-word results on two of the test cases. Each of these figures is split into four parts: a synopsis of the video motion, a plot showing where in the video clip the 10 most informative flow-words for classification originated (out of 4000 possible flow-words from this vocabulary), the most-informative frame marked with all the instances of any of these 10 highly-weighted words, and then finally the top four most informative flow-words for the given video clip.

In Figure 4, the example comes from one of the videos which was uploaded to YouTube after being time-reversed, and the synopsis images show a boy emerging from a pool with an inverse-splash. The motion of the water in this splash is the strongest indicator in this clip that time is running in the backwards direction. In Figure 5, time runs in the forwards direction as usual, and the synopsis shows a steam train gradually approaching the camera. The gentle upwards divergence, arising from the plume of steam from the train, is a good indicator of Time’s Arrow, and interestingly, the most-useful words for capturing this appear peri-

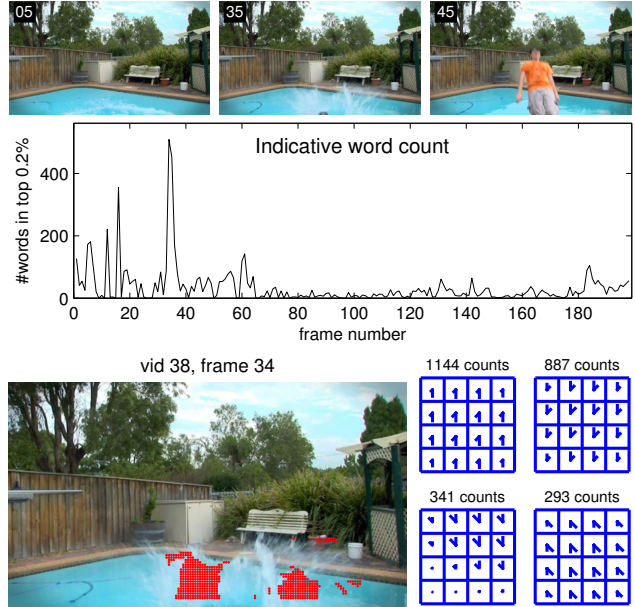


Figure 4. Video from the test set, correctly classified as being backwards in time. The most informative point in the video is the few frames around 34 (shown), where there is an inverse splash of water. Red dots here mark the locations of all strongly-backwards flow-words. This type of motion (shown in the four flow-word grids) was learnt to be strongly indicative of a backwards time direction.

odically in the video clip as the rhythm of the train’s motion causes the fumes to billow in a regular way.

Overall, there is a very slight tendency for flow-words describing forwards-time videos to have a higher divergence score than those describing backwards videos. Correlations scores between model weight and flow-word divergence are small but consistent, with models learnt on the three splits having correlation coefficients of 0.329, 0.033 and 0.038 respectively.

4. Motion-causation method

The flow-words motion above considered how we could examine simple motion patterns, but not how those patterns related to one another. To explore motion relations, we now sketch a method that considers motions *causing* other motions. Our starting hypothesis is that it is far more common for one motion to cause multiple motions than for multiple motions to collapse into one consistent motion. For instance, the cue ball during a snooker break-off hits the pack of red balls and scatters them, but in a reverse-time direction, it is statistically highly unlikely that the initial conditions of the red balls (position, velocity, spin etc) can be set up exactly right, so that they come together, stop perfectly, and shoot the cue ball back towards the cue.

To exploit this observation, we simply look at regions in

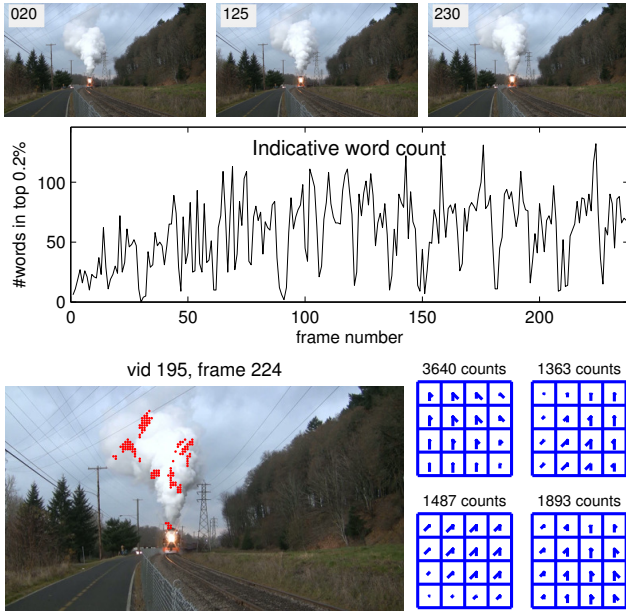


Figure 5. Video from the test set, correctly classified as being forwards in time. The most informative forwards-time words refer to the motion of the billowing steam, and occur periodically as the steam train puffs. Red dots marked on frame 224 show the locations of the most-forwards 10 flow-words in this frame.

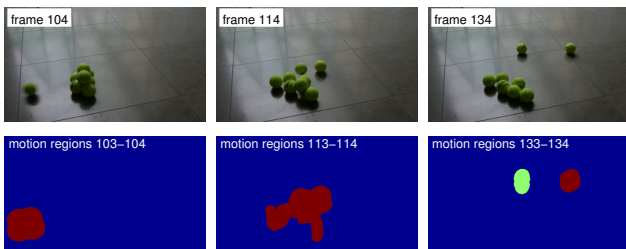


Figure 6. Three frames from one of the Tennis-ball dataset sequences, in which a ball is rolled into a stack of static balls. Bottom row: regions of motion, identified using only the frames at t and $t - 1$. Notice that the two rolling balls are identified as separate regions of motion, and colored separately 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.

the video from frame to frame where differences are appearing, which we assume are due to object motion. Smoothed regions of motion in one frame are compared to similar regions in the previous frame. These motion regions are illustrated in Figure 6. Broadly speaking, we expect more occurrences of one region splitting in two than of two regions joining to become one, in the forwards-time direction.

Implementation outline: We warp the image at $t + 1$ I_{t+1} into the frame of I_t , using pixels from I_t to fill any regions not present in I_{t+1} . This yields a warped image W_{t+1} . The difference $|I_t - W_{t+1}|$ now highlights mov-

ing areas. A smooth-edged binary mask of image motion is made by summing this difference over color channels, resizing down to 400 pixels, convolving with a disk function and then thresholding. The different regions in the motion mask are enumerated. Where one or more regions at time t intersect more than one region each at time $t - 1$, a violation is counted for that frame pair, since this implies motion merging. We count the violations for each time direction of a sequence separately, with a variety of parameter settings. We used three different threshold-radius pairs: radii of 5, 6.6 and 8.7 pixels, and corresponding thresholds of 0.1, 0.01 and 0.056, where image intensities lie between 0 and 1. Finally, we trained a standard linear SVM using the violation counts as 6d features (two time-directions; three parameterizations).

Experimental results: Results on splits A to C were 70%, 73% and 73% respectively. This is weaker than the flow-words method, and performance drops to just below 60% overall if only the first radius-threshold pair is used (*i.e.* a 2d decision space instead of 6d).

While the accuracy of the motion-causation approach is weaker, it is at the same time computationally much simpler and quicker to compute than the flow-words method. There is also the chance that it may complement the capabilities of the flow-words method well, because the motion-causation considers spatial location of motion, and comparison of motion between neighboring frames, whereas the flow-words method at present considers only snapshots of motion for each frame separately. Of the 50 videos that this system mis-classified, only 10 overlapped with the mis-classified set from the flow-words method.

On the Tennis-ball dataset, the motion-causation method also achieved 12/13 correct classifications; the mis-classified sequence was different to that mis-classified by the flow-words method.

5. AR method

Researchers have recently studied the question of measuring the direction of time as a special case of the problem of inferring causal direction in cause-effect models. Peters *et al.* [12] showed that, for non-Gaussian additive noise and dynamics obeying a linear ARMA (auto-regressive 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, and Peters *et al.* [12] used it to successfully analyze the direction of time for EEG data. Intuitively, this insight formalizes our intuition that changes (noise) added to a process at some point of time influences the future, but not the past.

Here we consider the special case of AR models (*i.e.* no moving average part); however, we deal with vector-valued

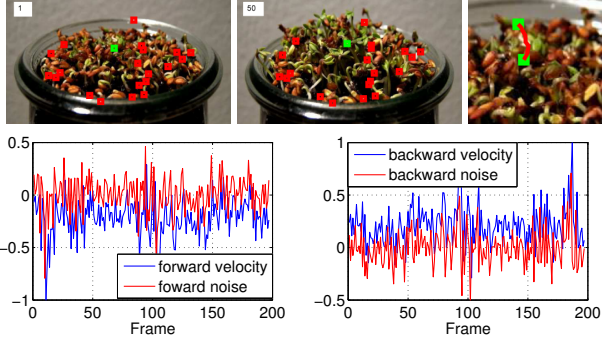


Figure 7. Overview of the AR method. Top: tracked points from a sequence, and an example track. Bottom: Forward-time (left) and backward-time (right) vertical trajectory components, and the corresponding model residuals. Trajectories should be independent from model residuals (noise) in the forward-time 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.

time series, which strictly speaking goes beyond the validity of the theoretical analysis of Peters *et al.* [12]. We found that an order of two worked well in our setting. In a nutshell, we model the time series’ next value as a linear function of the past two values plus additive independent noise.

The assumption that some image motions will be modeled 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). This process is illustrated in Figure 7.

The independence testing follows the work of [12], and is based on an estimate of the Hilbert-Schmidt norm of the cross-covariance operator between two reproducing kernel Hilbert spaces associated with the two variables whose independence we are testing. The norm provides us with a test statistic, and this in turn allows us to estimate a p-value for the null hypothesis of independence. If the p-value is small, the observed value of the norm is very unlikely under the null hypothesis and the latter should be rejected. Ideally, we would hope that the p-value should be small in the backward direction, and large (*i.e.* significantly bigger than 0) in the forward direction.

Implementation outline: We analyze the motion of a set of feature points extracted by KLT trackers [11, 20], running tracking in both forward and backward directions. For each tracked point, velocities are extracted, and a 2D AR model is fitted. We then test the independence between the noise and velocity to determine Time’s Arrow at the trajectory level.

Inferring causal direction of AR process is only possible when the noise is non-Gaussian, and when noise in only one

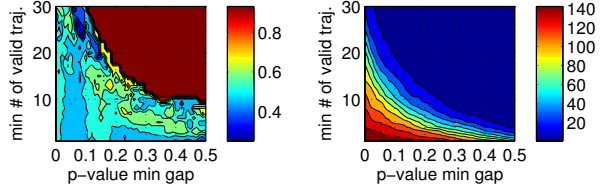


Figure 8. Left: Accuracy of AR model classifications over the 2D parameter space. Right: number of videos accepted. The AR model is particularly effective for small numbers of videos; as we force it to classify more, the performance falls to near chance level.

temporal direction is independent. We define a *valid trajectory* to be one which spans at least 50 frames, for which noise in at least one direction is non-Gaussian as determined by a normality test, and for which the p-value test in one time-direction gives $p < 0.05$ whereas in the other it gives $p > 0.05 + \delta$ for some minimal gap δ (*i.e.* exactly one direction fails the null hypothesis test).

All valid trajectories are classified as forward or backward according to their p-value scores. Ideally, all valid trajectories for one video should imply the same direction of time, but in practice, tracking can be noisy in a way that violates the time-series model assumption. For this, we reject the videos with fewer than N valid trajectories, where $N \geq 1$. We classify the *accepted videos* by a majority vote among the valid trajectories. We thus get a binary classifier (with the possibility of rejection) at video level. While hypothesis testing is used to classify single trajectories, the overall procedure is not a hypothesis test, and thus issues of multiple testing do not arise. The hypothesis testing based trajectory classifiers can be seen as weak classifiers for video and the voting makes a strong classifier.

Experimental results: The desirability of using a large gap δ in the p-value test, and requiring many valid trajectories per video must be traded off against the need not to reject excessively many videos. For large values $(\delta, N) = (0.24, 13)$, only 5 videos are accepted, but 4 are classified correctly (80%). For lower parameter values, $(\delta, N) = (0.15, 4)$, 101 videos are accepted, of which 58 are correctly classified (58%). Figure 8 shows more of this trade-off of between accuracy and number of videos classified.

6. Discussion

We have addressed a new and very fundamental problem of telling which direction time is flowing in a video, and presented three complementary methods for determining this direction. Our results indicate that the statistics of natural videos are not symmetric under time reversal: for if they were, then for each video, there would be an equally likely video where time is reversed, so there would be no way to solve the classification problem better than chance

(assuming our training sample is unbiased / representative of the prior).

Of course, each method can be improved. For example, the flow word method performed well above chance on this video set, but its performance can doubtlessly be improved by extensions to use multi-resolution in both space and time. State of the art features developed for other tasks with a temporal dimension, such as actions and activity recognition, could also be employed [8, 23].

The causality method captures an essential aspect of going forward in time, and can be extended to longer temporal intervals – for example to capture events that are causal but delayed, e.g. a tree branch oscillating some time after a person has brushed past it.

Similarly, the AR method as applied in our experiments is based on rather specific assumptions (2nd order linear AR model with independent non-Gaussian noise). If these assumptions hold true for a given video, we can expect a large p-value in the forward direction and a small one backward, and thus a big difference in p-values. This case turns out to be rare in our videos, but if it occurs, the method works rather well. If not, then both p-values should be small, and their difference is also small. Our method will reject such cases – rightly so, for if it is forced to decide these cases, then we find performance to be almost at chance level.

More generally, we have deliberately concentrated on low-level statistics in this work, but priors and classifiers can also be developed for high-level semantic information (like the fact that animals move head-forwards) to complement the low-level analysis.

Acknowledgements. This work was supported in the UK by ERC grant VisRec no. 228180, in China by 973 Program (2013CB329503), NSFC Grant no. 91120301, and in the US by ONR MURI grant N00014-09-1-1051 and NSF CGV-1111415.

References

- [1] C. Ballester, V. Caselles, J. Verdera, M. Bertalmo, and G. Sapiro. A variational model for filling-in gray level and color images. In *ICCV*, 2001. 2
- [2] T. Brox, A. Bruhn, N. Papenberger, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. Springer-Verlag, 2004. 4
- [3] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. Comput.*, 31(4), 1983. 1
- [4] T. Dekel Basha, Y. Moses, and S. Avidan. Photo sequencing. In *ECCV*, 2012. 2
- [5] K. G. Derpanis, M. Lecce, K. Daniilidis, and R. P. Wildes. Dynamic scene understanding: The role of orientation features in space and time in scene classification. In *CVPR*, 2012. 3
- [6] D. J. Field. Wavelets, vision and the statistics of natural scenes. *Phil. Trans. R. Soc. Lond. A*, 357, 1999. 2
- [7] J. Fiser and R. N. Aslin. Statistical learning of higher-order temporal structure from visual shape sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(3), 2002. 2
- [8] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal Localization of Actions with Actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11), Mar. 2013. 8
- [9] R. Kemp, G. Pike, P. White, and A. Musselman. Perception and recognition of normal and negative faces: the role of shape from shading and pigmentation cues. *Perception*, 25, 1996. 1
- [10] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 4
- [11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, 1981. 7
- [12] J. Peters, D. Janzing, A. Gretton, and B. Schölkopf. Detecting the direction of causal time series. In *ICML*. ACM Press, 2009. 6, 7
- [13] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40, 2000. 2
- [14] H. Price. *Time's Arrow and Archimedes' Point: New Directions for the Physics of Time*. Oxford University Press, 2007. 1
- [15] V. S. Ramachandran. *The Perceptual World*, chapter Perceiving shape from shading. W. H. Freeman, 1990. 1
- [16] H. Reichenbach. *The Direction of Time*. Dover Books on Physics, 1971. 1
- [17] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *CVPR*, 2005. 2
- [18] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. In *ICML*, New York, NY, USA, 2012. Omnipress. 2
- [19] J. H. Searcy and J. C. Bartlett. Inversion and processing of component and spatial-relational information of faces. *Journal of Experimental Psychology: Human Perception and Performance*, 22, 1996. 1
- [20] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, 1991. 7
- [21] P. Tuisku, T. K. Pernu, and A. Annala. In the light of time. *Proc. R. Soc. A*, (465), 2009. 2
- [22] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010. 4
- [23] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 4, 8
- [24] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *CVPR*, 2004. 2
- [25] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011. 2