Numerical Algos - Higher Order SVD and Dynamic Textures

Santripta Sharma

December 18, 2024

1 Extending the Singular Value Decomposition

In this course, we have seen how the properties of the singular value decomposition can be used to obtain a low-rank approximation of a matrix. We have also seen how this can be applied to a variety of problems, such as solving linear systems (either exactly or in a least-squares sense), computing the principal components of a dataset, compressing images, and even for obtaining a planar approximation of a map of cities.

As powerful as the SVD is, it falls short in problems where the data representation itself is of higher order than a matrix. For example, a common problem in signal processing is to analyse the evolution of a multi-dimensional signal, such as an image, over time [4]. In this case, the data is most naturally represented by a tensor of order 3, where the regular SVD is not directly applicable. This is where the Higher Order SVD, (sometimes called the Tucker Decomposition), comes into the picture.

1.1 Tensors

Before we dive into the HOSVD, I'm going to set up some notation for tensors. A tensor of order d is a d-dimensional grid, where each cell contains a number. Under this understanding, a vector is a first-order tensor, and a matrix is a second-order tensor. An example of the third order tensor is what I mentioned earlier, video data, where the first two indexing dimensions correspond to spatial dimensions, and the third corresponds to time. These "indexing dimensions" are idiomatically called modes in the context of tensors. In statistical modelling, tensors of even higher orders are used to deal with higher order moments, analogously to the covariance matrix (a second order tensor dealing with the second central moment) used for PCA [4].

Following the notation used in [1], I will denote tensors using calligraphic letters, such as $\mathcal{X}$. Further, since I'm interested in the specific application of the HOSVD to the analysis of dynamic textures, I will restrict my discussion to third-order tensors, which sufficiently model the data for this case. A third order tensor, then, can be represented as $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, where I, J, K are the dimensions of the tensor along the three indexes (two spatial and one temporal, in our case). In general, a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_p}$ is of order p.

1.1.1 Unfoldings

Tensors may be "unfolded" along any one particular dimension, yielding a matrix. For instance, we can unfold a video tensor along the temporal mode (of size K) to obtain a matrix of size $K \times IJ$, where each row corresponds to a frame of the video (unfolded into a vector). This is called the mode-3 unfolding of the tensor, since K is the third mode. We can similarly obtain unfoldings along the other modes. We write the mode k unfolding of a tensor $\mathcal{X}$ as $\mathcal{X}_{(k)}$.

1.1.2 Tensor-Matrix Products

A tensor-matrix product is defined on a mode and yields a tensor of the same order as the original tensor. Consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_p}$, and a matrix, $\mathbf{B} \in \mathbb{R}^{J \times I_k}$. Since this matrix has the same number of columns as the kth mode of the tensor, we can compute their k-mode product, which is a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times \cdots \times I_p}$.
We say $\mathcal{C} = \mathcal{A} \times_k \mathbf{B}$, defined as follows:

$$(\mathcal{C})_{i_1 i_2 \ldots i_{k-1} j_k i_{k+1} \ldots i_p} = \sum_{i_k=1}^{I_k} (\mathcal{A})_{i_1 i_2 \ldots i_p} (\mathbf{B})_{j_k i_k}$$

To approach what is happening here, consider the matrix-matrix product. In the matrix-matrix product, we take the dot product of each row of the first matrix with each column of the second matrix. In the tensor-matrix product, we are doing the same, but along the kth mode of the tensor. This is a generalization of the matrix-matrix product to tensors.

Fortunately, although I have provided this definition in much generality, we only care about third order tensors, and can greatly simplify our mental model accordingly. As [1] shows, this can be written as a matrix product between the mode-k unfolding of the tensor and the matrix:

$$\mathcal{C}_{(k)} = \mathbf{B}\mathcal{A}_{(k)}$$

(which also answers the question of how **exactly** this product relates to the standard matrix product).

1.1.3 h-rank

The h-rank of a p-order tensor $\mathcal{A}$ is the tuple $(r_1, r_2, \ldots, r_p)$, where r_i is the rank of $\mathcal{A}_{(i)}$, the mode-i unfolding of $\mathcal{A}$. There is another notion of the rank of a tensor, but I will omit it, since it is not relevant for the coming discussion.

1.2 HOSVD

We've finally developed enough machinery to discuss the Higher-Order SVD. As the SVD decomposes a matrix into a scalar matrix $\mathbf{\Sigma}$ and two orthogonal matrices $\mathbf{U}, \mathbf{V}$, the HOSVD decomposes a tensor into a core tensor $\mathcal{S}$ and a set of p orthogonal matrices $\mathbf{U^{(i)}}$. The core tensor contains the most important information about the tensor, while the orthogonal matrices contain the basis vectors for each mode of the tensor. $\mathcal{S}$ may not be diagonal, however. The HOSVD is defined as follows:

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_p \mathbf{U}^{(p)}$$

As in the case of the SVD, this decomposition works for any arbitrary tensor (although the computation may be poorly conditioned in some cases). We can truncate the decomposition by keeping only the first r_i columns of each orthogonal matrix, and the first $r_1 \times r_2 \times \cdots \times r_p$ block of the core tensor, to obtain a rank $(r_1, r_2, \ldots, r_p)$ approximation of the original tensor.

It is important to note that, unlike the traditional SVD for matrices, this is not the **best** rank $(r_1, r_2, \ldots, r_p)$ approximation of the tensor. However, as [4] shows, it is a reasonably good approximation in most cases, with the approximation error being very small when compared to alternate, more computationally complex methods.

In order to compute the higher-order SVD, we use the method described in [1]:

- For $i \in \{1, 2, \ldots, p\}, \mathbf{A}_{(i)} = U^{(i)}\Sigma V^{\dagger}$.

  That is, we can compute the orthogonal matrices by performing the SVD on the mode-i unfolding of the tensor.
- Invert the orthogonal matrices to obtain the core tensor: $\mathcal{S} = \mathcal{A} \times_1 \mathbf{U}^{(1)t} \times_2 \mathbf{U}^{(2)t} \times_3 \cdots \times_p \mathbf{U}^{(p)t}$.

2 Dynamic Textures

Dynamic textures are a class of videos which are characterized by some kind of spatio-temporal regularity [2]. Classical examples include videos of clouds, waves, open flames, etcetera. These videos are often used in computer vision and machine learning to test algorithms for video analysis, such as object tracking, action recognition, and video compression.

One way to model dynamic textures is by modeling them as linear dynamic systems. [2] models these textures as auto-regressive moving average processes:

$$\begin{cases} x(t+1) = Ax(t) + Bv(t) \\ y(t) = \phi(x(t)) + w(t) \end{cases} \tag{1}$$

Where $x(t)$ is the original signal at time t, $\phi(\cdot)$ a spatial filter such that the image at time $t, I(t) = \phi(x(t))$ (typically the id function), and $y(t)$ is a noisy measurement of our signal (via a camera or whatever). $v(t)$ and $w(t)$ are respectively the model and measurement noise terms.

As [1] suggests, a common way to build these models is by taking a dynamic texture, which is an $I \times J \times K$ tensor, and unfolding it along the temporal mode to obtain a matrix of size $K \times IJ$. We can then use the classical SVD to obtain a low-rank approximation of the matrix, and hence a low rank model of the system's behaviour. The problem with this, of course, is that the flattening of the tensor along the temporal mode leads to a loss of spatial information in the model, which means we are not capturing spatial relations as effectively as we could be.

The HOSVD provides a solution to this problem. By decomposing the tensor into a core tensor and orthogonal matrices, we can obtain a low-rank model of the dynamic texture which retains spatial relations. Then, we can try synthesising new dynamic textures by running the model forward in time starting from random noise.

Figure 1: Ten subsequent frames from three flame videos after preprocessing

2.1 Dataset

I'm using the DTDB[3] dataset for all experiments in this assignment. The dataset contains a variety of dynamic textures, although I am only interested in working with flames for the time being.

For preprocessing, I first extract video frames, resize them to 128×128, convert them into grayscale, and black-out all values below a threshold in order to remove noise. I then stack these frames into a third-order tensor, where the first two modes correspond to height and width, and the third mode corresponds to time. This tensor is then used for all further analysis. Figure 1 shows a sample of a few post-processed frames.

3 Modelling & Synthesis

I follow the procedure laid out in [1], consisting of the following steps:

- Mean-center the tensor about the temporal mode (computing a mean tensor $\mathcal{M}$ along the temporal mode).
- Perform the HOSVD on this tensor, and truncate the decomposition to a rank (r_1, r_2, r_3).
- Work on the $\mathbf{X} = \mathbf{U}_{r3}^{(3)}$ matrix, since this contains temporal information about the dynamics of the system. If $\mathbf{x_j}$ is the jth row of $\mathbf{X}$, we want to find an $\mathbf{H}$ such that $\mathbf{x_{j+1}} = \mathbf{H}\mathbf{x_j} + \mathbf{e_{j+1}}$, where e_{j+1} is some noise. We can do this by performing a least squares regression on the matrix.
- We also need to model the noise e_{j+1}. We can do this using a multivariate gaussian, estimating the input noise covariance matrix $\mathbf{Q}$ from the observed residuals $(\mathbf{x_{j+1}} - \mathbf{H}\mathbf{x_j})$. We then normalise this into $\mathbf{G}$, such that $\mathbf{Q} = \mathbf{G}\mathbf{G}^\mathbf{T}$ (Cholesky).
- Finally, our model is parametrised by the truncated (to rank r_i) orthogonal $\mathbf{U}_{\mathbf{r_i}}^{(\mathbf{i})}$ matrices, core $\mathcal{S}_{\prod r_i}$ tensor, matrices $\mathbf{H}, \mathbf{G}$, and the mean-centering subtraction tensor $\mathcal{M}$.

Once we acquire the model following these steps, we can then drive the model using the following equation:

$$\begin{cases} \mathbf{x(t+1)} = \mathbf{H}\mathbf{x(t)} + \mathbf{G}\mathbf{v(t)} \\ \mathcal{Z}(t) = \mathcal{S}_\mathbf{r} \times_1 \mathbf{U}_{r_1}^{(1)} \times_2 \mathbf{U}_{r_2}^{(2)} \times_3 \mathbf{x(t)} + \mathcal{M} \end{cases} \tag{2}$$

Here, $\mathcal{Z}(t)$ is the synthesised tensor at time t, $\mathbf{x(t)}$ is the state of the system at time t, and $\mathbf{v(t)}$ is a gaussian random vector.

4 Experiments

For all these experiments, I've used truncation to the same rank across all modes (i.e., $r_1 = r_2 = r_3 = r$). This is done to keep things simple, but [1] advises that different rank can be chosen for different modes depending on how much variation the dynamic texture shows along each mode. For instance, a dynamic texture which changes rapidly in time (rapidly blinking point light source, for eg) but not much in space would have a higher rank along the temporal mode than the spatial modes, whereas a texture which changes more rapidly in the vertical direction than the horizontal (blades of grass in the wind) would have a much higher vertical spatial rank than the horizontal and temporal ranks.

Frames	Rank	Time (s)
272	10	4.5
272	50	5.5
272	100	7
272	128	10.5
284	10	5.9
284	50	6.6
284	100	9.6
284	128	14.5
299	10	5.4
299	10	5.9
299	10	9.1
299	10	13.8

Table 1: HOSVD computation time for different approximations

Scene	Mean Residual	Max Residual	Time (s)
1	1.058e-5	0.0025	3.7
2	1.690e-5	0.0038	4.1
3	1.387e-5	0.0026	3.9

Table 2: HOSVD synthesis time & mean residuals for different approximations

4.1 Reconstruction

At first, we just perform the HOSVD on our tensors and see how well they are reconstructed by truncations to different ranks. I've also indicated the time taken for each rank in Table 1. Figures 2, 3, and 4 show the original and reconstructed frames for three different videos. As we can see, the reconstruction is reasonably good, although a bit noisy (which we correct for within our model).

4.2 Synthesis

Similarly, we try "training" a model on the tensors for each of the shown flame videos, and then synthesise new frames by running the model forward in time, sampling $\mathbf{v}(\mathbf{t})$s from the standard multivariate gaussian.

For synthesis, I've used a rank of 100 for all modes, and the results are shown in Figures 5, 6, and 7. As we can see, the synthesised frames are distinct enough from the original frames to be considered new frames, due to the randomly sampled noise, but they still retain the general dynamics of the flames in the original flames.

In Table 2, I've also indicated the mean residuals and the time taken for synthesising 100 frames. The mean residuals are quite low, indicating that the linear model is quite good at capturing the dynamics of the flames. Furthermore, the technique is also quite efficient, able to synthesise 100 frames in under 5 seconds, which means it is quite feasible to use this technique for real-time applications.

What seems to be clear is that the system has a harder time modelling the more restrained "masked out" flames (firewood etc. removed by threshold) in scenes 1 & 2 than the more unrestrained blobby frame in scene 3. This is possibly because the masked regions in the first two scenes add spatial complexity, which would require higher spatial ranks to model effectively. Further postprocessing (or data preprocessing) may be able to get more realistic or visually appealing results, but I am quite happy with the results as they are, since we have at least effectively been able to synthesise a new flame sequence from a low-rank approximation of the original.

Figure 2: Reconstruction of a video with 272 frames, original on the bottom

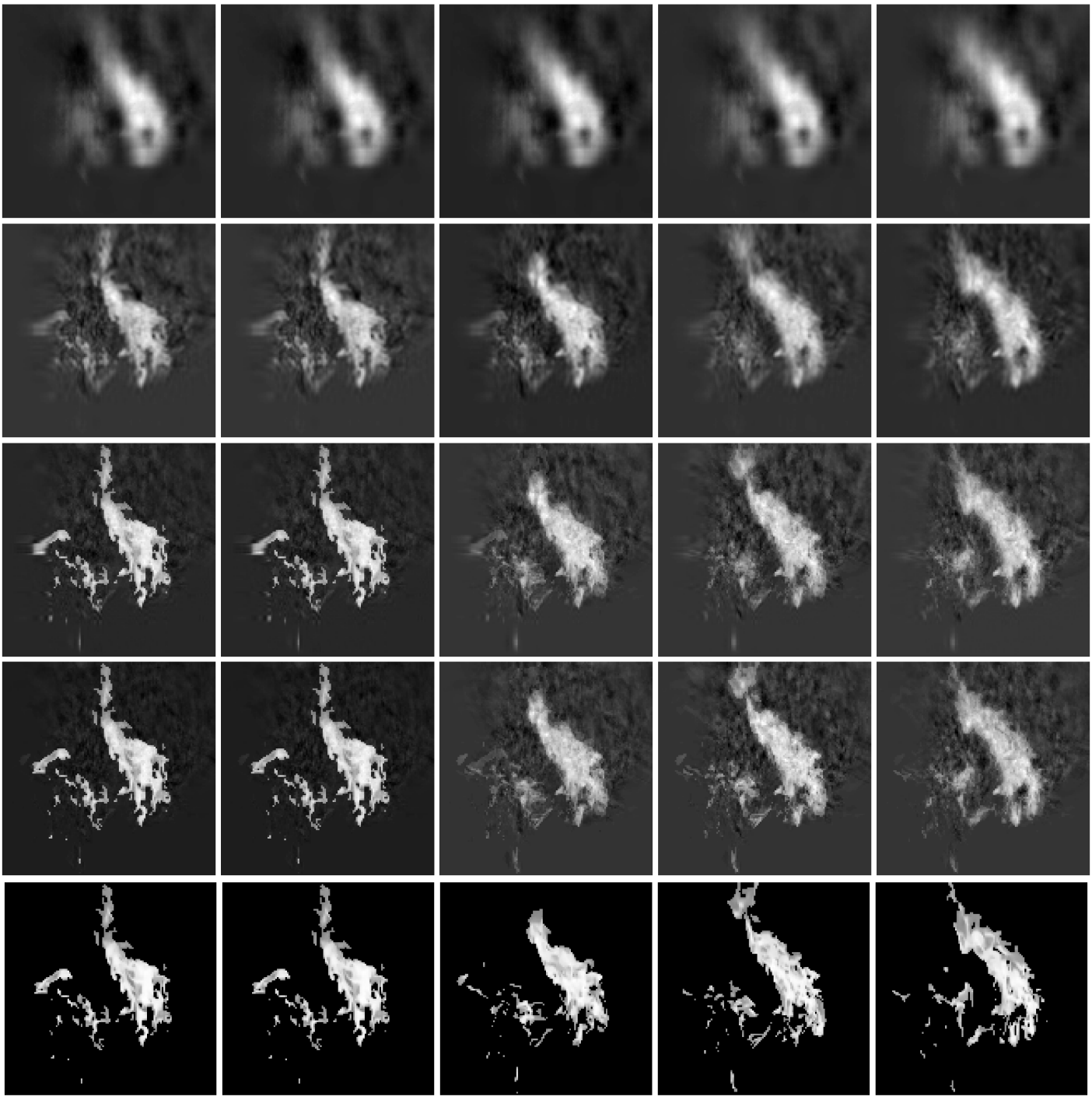

Figure 3: Reconstruction of a video with 284 frames, original on the bottom

Figure 4: Reconstruction of a video with 299 frames, original on the bottom

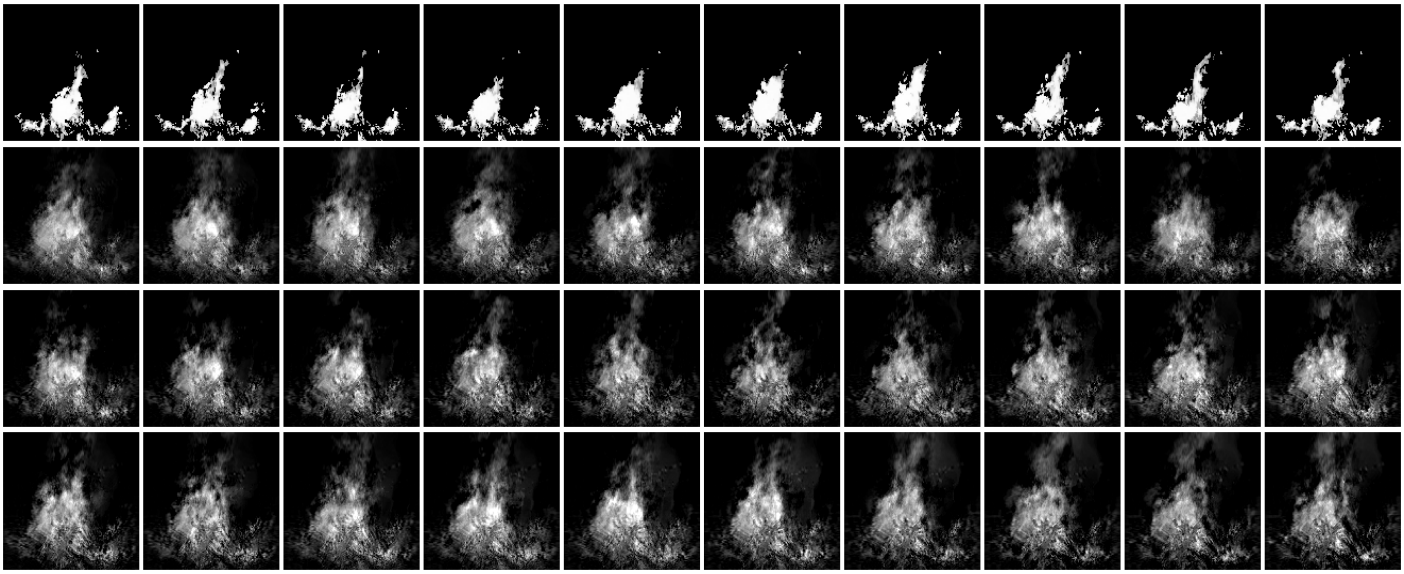

Figure 5: Synthesis results for scene 1, first row is original, rest are thirty frames from the synthesised video

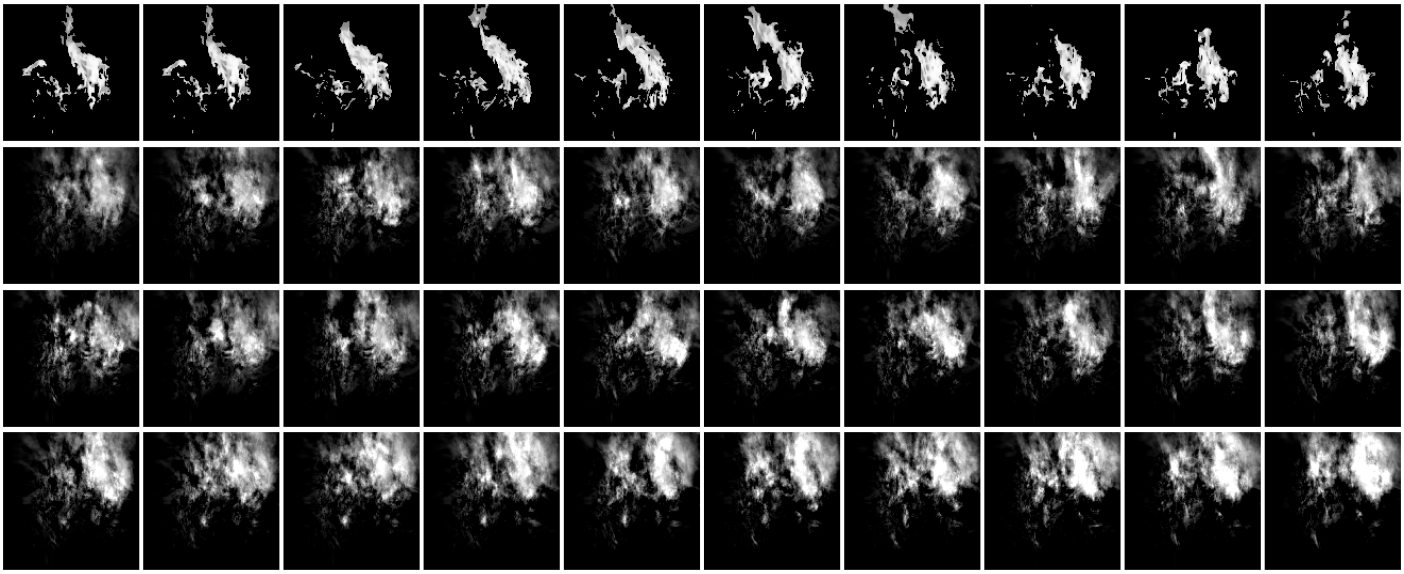

Figure 6: Synthesis results for scene 2, first row is original, rest are thirty frames from the synthesised video

Figure 7: Synthesis results for scene 3, first row is original, rest are thirty frames from the synthesised video

References

[1] Roberto Costantini, Luciano Sbaiz, and Sabine Susstrunk. Higher order svd analysis for dynamic texture synthesis. *IEEE Transactions on Image Processing*, 17(1):42–52, 2008.

[2] Gianfranco Doretto, Alessandro Chiuso, Yingnian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51:91–109, 02 2003.

[3] Isma Hadji and Richard P. Wildes. A new large scale dynamic texture dataset with application to convnet understanding. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XIV*, page 334–351, Berlin, Heidelberg, 2018. Springer-Verlag.

[4] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.