Generating Video Thumbnails using SVD

Matthew Kuzyk, [redacted] and Valentin Tsatskin, [redacted]

***Abstract*—In this report, video summary for the purpose of generating efficient thumbnails is discussed. It is implemented using Singular Value Decomposition to generate a heuristic for variance in visual information across frames in a video. The information in each frame is represented using a 3D histogram of the RGB colours. This heuristic is then used to try and group every frame based on smallest possible visual variance. A case study that would be applicable and useful to summarise in the real world is then tested against, and its clustering and performance results are analysed. Performance and accuracy improvements are suggested, as well as interesting observations garnered from altering the data and algorithm.**

I. Background

Thumbnails are a collections of frames from a video that are meant to entice the viewer into opening the video. Often, sites let the viewer preview the video by viewing a collection of thumbnails summarising the video. Summarising video content using thumbnails is an important problem in the video sharing industry. It is the subject of extensive effort and research. "Video thumbnails are often the first things viewers see when they look for something interesting to watch. ... Better thumbnails lead to more clicks and views for video creators" [2]. For a site like YouTube, clicks on videos mean money, due to the revenue being generated on ads seen while watching a desired video. For these reasons, the thumbnails of a video should be the best possible summary they can be.

One way to summarise a video would simply be to sample the frames at a constant rate and use these as the thumbnails. However, much of the video might be very visually similar, with short, yet important visual scenes in between that will not be shown by this technique. The user would miss out on important information in the video while being subjected to many similar scenes.

For this reason, it is very useful for video content providers to automatically figure out which parts of a given video are most visually distinct and summarise them for the user. To do this, the techniques in the paper "Video summarization and retrieval using singular value decomposition" will be relied upon [1].

II. Relevance of Linear Algebra

Linear algebra is relevant to the problem chosen since it can be used to determine visual content in an image. For instance, in principle component analysis, a set of datapoints, will be projected onto a space with an orthogonal eigenvector basis. The distances to the axis represent the variance from each data point to the axis [3]. The application here is very similar, the data will represent spatial colour information about the video frames, and their variance will represent their similarity. In fact, PCA can be implemented using Singular Value Decomposition (used later in this report). From this, it should be obvious that linear algebra is very useful for classifying visual similarity of images.

III. Case Study

For the case study, a clip from a talk show was chosen. This is because on a talk show, there might be many minutes of the host and guest sitting still and talking, with a short interlude skit outdoors. The user, who is browsing for videos, will want to see a summary of what is in the video before opening. Though it is certainly relevant to show thumbnails of the host and guest, each new frame of this shows very little new visual information, and is thus redundant to the user. Because of this, clips from a talk show (The Eric Andre Show) were used that include multiple hosts, guests, and sketches. The video will be segmented into visually similar clusters and samples from each of these clusters will be shown. This is quite relevant for video content distribution, since there are many talk show clips on YouTube, and it is important for the user to see a summary of what happens in that clip for them to even consider opening it.

IV. Mathematical Solution

A. Feature-frame matrix creation

First, the feature space for the entire video needs to be created. This will represent the visual information in the video. To start, the video must be decomposed into frames. Note that it is highly unlikely that there is relevant visual information to the user in all of the 30-60 frames that appear every second in the video. For this reason, a more efficient sample rate of 5 frames per second was used.

A histogram matrix representing the colour in the frame was needed. Taking a matrix representing each pixel's intensity in the RGB colour space would not be feasible as the size of the matrix would be too large. A 720x480 pixel frame, for instance, produce a matrix with 1,036,800 elements. In order to reduce the size of the matrix involved but keep spatial and colour information in-tact, a dimensionality reduction process is proposed.

Each frame of the video is split up into 3x3 blocks and a 3D histogram approach for the RGB colours is applied. This matrix represents the frequencies of all of the possible 255^3 colour permutations a pixel could have. This is still too large to work with so instead each channel is grouped into 5 bins, meaning 5 is the most intense colour value of the channel and 0 the least. This yields 5^3 possible permutations representing every colour. The *pixel colour vector* is generated, where the value in each field represents the frequency of the colour permutation in a block. Concatenating each pixel colour vector from all blocks in order from top left to bottom right yields a vector of length 1125. These vectors will be used later in the Single Value Decomposition. Using ordered histograms

of pixel blocks will be better than using that of the whole image, since two very different images might have a similar number of colours spread out across different locations. This 3x3 block approach will preserve information about the colour makeup in each location of the frame.

The matrix A will represent visual information across the entire video. Every column entry is the pixel colour vector of a given frame, therefore, the number of rows will be 1125 and the number of columns n will be the number of frames collected from the video.

B. Singular Value Decomposition

As mentioned earlier, the eigenvectors of the matrix representing the visual information (A) will be needed to determine image variance and similarity.

To do this, the matrix is then decomposed using Singular Value Decomposition (SVD). SVD is necessary since it will never be guaranteed that the number of total bins m will equal the number of frames n, which would make the matrix diagnosable and thus allowing standard Eigenvalue Decomposition (EVD). The SVD is done as follows:

$$A = USV^T \tag{1}$$

Where the matrix V^T has rows made up of the eigenvectors of A^TA and the columns of the U matrix represent the eigenvectors of AA^T. The matrix S consists of singular values along the diagonal σ_i relating each v_i row and u_i column, where the corresponding eigenvalue is σ_i^2. The full SVD algorithm was chosen, although is not necessary since the extra orthogonal u_i vectors will not often be used. Also note that most videos will not contain every possible colour permutation, and because of this, the A matrix often ends up being sparse. Thus, more efficient SVD algorithms meant for sparse matrices could be used [1].

The SVD is important because it defines the mapping between the 1125 dimensional feature space seen in the previous subsection and the linearly independent r-dimensional singular vector space. It will map each frame A_i into a new vector, ψ_i which is defined as follows.

$$[\psi_1 \quad \cdots \quad \psi_i \quad \cdots \quad \psi_n] = V^T \tag{2}$$

Thus, the vectors ψ_i are the rows of the V matrix. The ψ_i vectors can be thought of as the *principal direction* of the visual information. It is what describes the general look of each frame before it is scaled by its corresponding singular value. A_i will also map its rows, which represent the occurrence count of a given pixel value across all frames, into the vector ϕ (the columns of U).

An important note: this SVD decomposition requires that $m > n$ for matrix A, meaning there must be no more than 1125 frames. If, however, $m < n$, then the roles of the ϕ and ψ matrices are switched entirely [1]. Note, for the remainder of the report, it will be assumed that $m > n$, so the steps outlined still apply.

C. Clustering

Using the results from the SVD, the first steps in summarising the video can begin. The first useful metric is the norm of each ψ_i calculated above. The norm of a vector is simply the inner product between the vector and itself, $\langle \psi_i, \psi_i \rangle$. Only the columns up to the $rank(A)$ need to be considered since the rest will contain redundant information. Since real numbers are being dealt with here, this inner product is simply:

$$\|\psi_i\| = \sqrt{\sum_{j=0}^{rank(A)} v_{ij}^2} \tag{3}$$

This metric is what defines the variability of a video segment. Suppose that the matrix A has many very similar frames as its columns. If the frames are all similar, they will often be linearly dependent. The more linearly dependent columns there are in A, the lower the variance will be in the images. Since eigenvectors summarise the visual information across frames, if there is a lot of similarity between many frames, the smaller the contribution to the overall visual information each of those similar frames will have [1]. If A has rank of n, the norm will be 1. If it has a very low rank, the norm will be small. Therefore, the total amount of visual content in a video segment S_i defined by A is simply equal to:

$$CON(S_i) = \sum_{i=0} \|\psi_i\|^2 \tag{4}$$

This metric can then be used to group the video segment into clusters of visual information.

V. Clustering

Next, the information will be clustered. First, the minimum distance of two vectors ψ_i and psi_j must be defined. It is the Euclidean distance between the two vectors weighted by singular values σ_k all the way from $k = 0$ up to a certain τ value.

$$D(\psi_i, \psi_j) = \sum_{i=0}^{\tau} \sigma_k (v_{ik} - v_{jk})^2 \tag{5}$$

This τ defines how many singular values are to be used in the metric. This boundary is to be determined based on empirical testing of training data. In the paper, the τ that works best for their purposes is a value of $\tau = 50$ [1]. This same value will be used in our testing. Next, the visual clustering can begin as follows:

1) For all unclustered ψ, sort them by norm in ascending order and select the one with the smallest norm
2) Use this vector to form the current cluster S_C, it will have an average internal distance $\bar{R}(S_C) = 0$ and a frame count $P_C = 1$
3) For the next remaining unclustered ψ_i, calculate its min distance to S_C, which is defined as:

$$d_{min}(\psi_i, S_C) = minD(\psi_i, \psi_k) \tag{6}$$

For every k vectors in S_C. D comes from equation 5.

Fig. 1: Sample frames from generated clusters

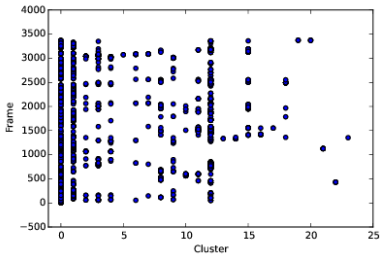

Fig. 2: Scatter plot of cluster number vs sequential frame number.

4) If $C = 1$, this is this first cluster, add the frame ψ_i to the cluster S_1 as long as $\bar{R}(S_1) = 0$ or $d_{min}(\psi_i, S_1)/\bar{R}(S_1) < 5.0$. If the average internal distance of S_C is 0, if $CON(S_C) < CON(S_1)$, or if $d_{min}(\psi_i, S_C)/\bar{R}(S_C) < 2.0$, then add ψ_i to the S_C
5) If the frame was added to the cluster S_C, then increment the frame count P_C by 1, and update the content of value $CON(S_C)$ using equation 4.
6) If the frame was added to the cluster, update the internal distance of the cluster as follows:

$$\bar{R}(S_C) = \frac{(P_C - 1)\bar{R}(S_C) + d_{min}(\phi_i, S_C)}{P_C} \quad (7)$$

7) Increment i, repeat steps 3-6 until there are no more unclustered ψ

VI. Results

All of the above steps were implemented using Python, the code can be found on GitHub [4]. Using a talk show clip, the sample frames from each of the clusters can be seen in Figure 1.

As can be seen in Figure 1, the algorithm performs quite well. It clusters differently depending on which guests appeared at the time, and also the type of situation as well (outdoors, indoors, etc). The actual cluster data can be seen in Figure 2. To interpret this, the axes must be understood: The x-axis is the cluster number, and the y-axis is the sequential frame number.

Interestingly, over the course of multiple tests, the first cluster was usually the worst. This makes sense given steps 2 and 4 in the clustering algorithm above. For the first cluster, after the first seed vector has been added, it will have an average internal distance $\bar{R}(S_1) = 0$, since there is only one vector inside (no distance from itself). For the next vector in step 4, since $\bar{R}(S_C) = 0$ and $C = 1$, it will automatically add this vector to the cluster. This means that no matter how dissimilar the first two frames analysed are, they will both be added to first cluster. Since this seeds the information for the rest of the frames that will be put in the cluster, it will classify poorly, since the initial frames are dissimilar. This can be seen in Figure 2 since the very first cluster groups almost every different type of frame across its history. It should not be that dense, the clusters with better defined gaps perform better. Increasing the number of bins is suspected to increase the accuracy of clusters, since there will be more colour precision. However, this increases the size of the matrix A at a cubic rate and can slow this down significantly. Another approach suspected to improve clustering would be to use randomised k-means clustering n times and picking the set of clusters which have the smallest amount of variance across clusters.

When the 3x3 window size is made smaller, it does not necessarily improve the accuracy of the clusters. In our testing, it made the clustering less accurate at a 9x9 window. To illustrate why this makes sense, consider somebody nudging their hand while in frame. If the windows are too small, the hand might exit one window into the next. This means that the histograms that the hand moved between will be completely different from what they were in the previous frame, even though the change for the whole image is slight.

VII. Conclusion

Using SVD to summarise videos is a feasible task. Combining this with user input, cluster A/B testing combined with click metrics, and machine learning, video content providers could use this to automate video summary and improve their clickthrough rate. Though the implementation designed was computationally taxing, as it took a couple of minutes to complete on longer videos, more efficient algorithms could be combined to run this process on videos at upload time with no noticeable impact to the user.

References

[1] Y. Gong and X. Liu,*Video summarization and retrieval using singular value decomposition*, NEC Laboratories of America

[2] W. Yang and M. Tsai, *Improving YouTube video thumbnails with deep neural nets*, Google

[3] G. Dallas, *Principal Component Analysis 4 Dummies: Eigenvectors, Eigenvalues and Dimension Reduction*

[4] https://github.com/MattyKuzyk/thumbnail-generator-svd