Key Frame Detection of Videos Using DCT and LSH

Yulong Li
yl4095@columbia.edu

This paper focuses on the problem of detecting key frames of a video. For simplicity, we consider only gray-scale videos, which can be represented by 2D matrices. We approach this problem as a locality-sensitive hashing (LSH) problem for matrices. For this end, we define a metric for matrices using discrete cosine transform (DCT) and the Frobenius norm. We prove the proposed metric is indeed permissible and demonstrate its effects with experiments. We then develop an algorithm based LSH scheme and this metric and lay out further experimental results.

1 Introduction

Video streaming has become increasingly popular recently. A practically useful task is to find the key frames of a video, which is a nearest neighbor problem under an appropriate metric. Our goal is to solve this probelm with sub-linear memory space and near linear update time, without much trade-off for accuracy. For this paper, we will consider gray-scale videos only, and thus the input of our task can be considered as a stream of 2-dimensional matrices.

An important component of the problem is the metric d of matrices. Human perception can easily identify the key frames of a video, or how different two frames are, but an appropriate metric can be hard to define mathematically. For example, the Frobenius norm between similarly looking frames can be very large due to small displacements or rotations. Usually, we need some kinds of exaustive search between two frames to determine wether they are similar or not. However, this can be computationally expensive. We approach the problem by transforming the frame to its frequency domain via the discrete cosine transform (DCT) [ANR74], which is less sensitive to small distortions. We demonstrate that the Frobenius norm after DCT is still a permissible metric. We then use experiments to show that this metric indeed concord with our intuitive definition of similar frames. DCT also admits a easy lossy compression scheme by simply dropping coeffcients for the higher-order terms.

We then approach the nearest-neighbor problem by devicing an appropriate family hashing functions that assigns each matrix to a representative bucket, and keep only one frame for each bucket. Our task can be thus forumlated as a locality-sensitive hashing (LSH) [IM98] problem:

Given the metric space $(\mathbb{R}^{n\times n}, d)$, a threshold $r > 0$, an approximation factor $c > 1$, find a function $h : \mathbb{R}^{n\times n} \to U$ such that

$$\begin{cases} \Pr[h(A) = h(B)] > p_1 & d(p,q) < r \\ \Pr[h(A) = h(B)] < p_2 & d(p,q) > cr \end{cases}$$

We'd like h to 1) make the gap between p_1 and p_2 large, 2) have a image consisting of elements with memory sub-linear in n, and 3) compute fast.

We combine our proposed metric and the LSH scheme proposed by Datar et al. [Dat04] to complete our algorithm for key frame detection.

2 Preliminaries

2.1 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) [ANR74] is similar to the discrete Fourier transform: it transforms a signal (an image) from the spatial domain to the frequency domain. The set of orthogonal basis for 2-d DCT ordered in terms of increasing frequencies can be visualized as pixels of increasing resolutions (Figure 1). Each entry of the transformed matrix represents the magnitude of a particular basis. Intuitively, higher frequencies terms can be omitted without much loss of the information (comparing to omit low frequency term). This intuition will be made formal in the sense of Least Square Approximation. We define the $C \in \mathbb{R}^{n\times n}$,

$$(C)_{i,j} = \Lambda(i)\sqrt{\frac{2}{n}}\cos(\frac{\pi}{2n}(i-1)(2j-1))$$

where $\Lambda(i=0) = 1/\sqrt{2}$ and $\Lambda(i \neq 0) = 1$. One can check that C is a orthornormal matrix, so we can trasform a matrix $X \in \mathbb{R}^{n\times n}$ to the basis of C with a usual change of basis. This is defined as 2-d DCT of X:

$$DCT(X) = CXC^T$$

Note that given a $Y \in \mathbb{R}^{n\times n}$, which is the result of 2-d DCT of X, we can recover X by a Inverse Discrete Cosine Transformation (IDCT): $IDCT(Y) = C^TYC = X$. This can also be understood as interpolating X with a polynomial over 2-variables and using Y as the coefficients:

$$P(u,v) = C^TYC = \frac{1}{n}(Y)_{1,1} + i,j + \frac{2}{n}\sum_{i=1}^{n}\sum_{i=1}^{n}(Y)_{i,j}\cos\frac{i(2u+1)}{\pi}\cos\frac{j(2v+1)}{\pi}$$

such that $P(i,j) = (X)_{i,j}$ We can define the least square approximation problem as: given a $m \in [1,n)$, find $\overline{Y}$ to form

$$P_m(u,v) = \frac{1}{n}(\overline{Y})_{1,1} + i,j + \frac{2}{n}\sum_{i=1}^{m}\sum_{i=1}^{m}(\overline{Y})_{i,j}\cos\frac{i(2u+1)}{\pi}\cos\frac{j(2v+1)}{\pi}$$

such that $\sum_{i=1}^{n}\sum_{j=1}^{n}(P_m(i,j)-(X)_{i,j})^2$ is minimized. It follows from the orthonormality of C that $\overline{Y} = Y_m$ where Y_m is the sub-matrix of Y with first m rows and columns. This gives us a very convenient way to keep a lossy representation of a matrix X with least information loss: we can simply keep the sub-matrix Y_m in the DCT basis. This proves to work fairly well in the later section.

2.2 Locality-Sensitive Hashing for Vectors

2.2.1 Locality-Sensitive Hashing (LSH)

For low-dimension data, a tree-like space partitioning data structure can efficiently solve the nearest neighbor problem. However, to find the exact nearest neighbor for higher-dimensional space, we are not able to improve too much comparing to a naive linear search. Locality-Sensitive Hashing is proposed to find a approximation of the nearest neighbor, which works pretty well comparing to the exact solution and enables much more efficient algorithms. The idea of LSH is to use several hash functions such that near neighbors have much higher probability of collisions.

A locality-sensitive hashing family [IM98] $\mathscr{H}$ is a set of functions defined for a metric space $(\mathbb{R}^d, d)$, two threshold values $r > 0, c > 1$, such that by choosing a function $h : R^d \to U$ in the family uniformly at random,

$$\begin{cases} \Pr[h(A) = h(B)] > p_1 & d(p,q) < r \\ \Pr[h(A) = h(B)] < p_2 & d(p,q) > cr \end{cases}$$

for some probabilities p_1 and p_2. Such a family $\mathscr{H}$ is called (r, cr, p_1, p_2)-sensitive. We note that LSH as the basis is inherently randomized, since a single deterministic hash function may perform badly on some adversarial distribution. Formally, $\mathscr{H}$ is useful when $p_1 > p_2$, so that we can amplify the gap by the following. For some integral parameters k, L, we define a function family $\mathscr{G} = \{g : \mathbb{R}^d \to U^k\}$ where $g(v) = (h_1(v), \cdots, h_k(v))$, where $v \in \mathbb{R}^d, h_i \in \mathscr{H}$. For each input v, we choose L functions $g_1, \cdots, g_L$ from $\mathscr{G}$ uniformly at random and search all buckets $g_1(v), \cdots, g_L(v)$. For each u in these buckets, we check up to $3L$ points if $d(v,u) < cr$. In this way, we can ensure the correctness of the algorithm with high probability. It is proven by Indyk, et al. [IM98] that, given a universe of N elements, by setting $k = \log_{1/p_2} N, L = n^\rho$ where $\rho = \frac{1/p_1}{1/p_2}$, the algorithm finds a correct neighbor (or report correctly the point has no neighbor) with a constant probability. Further, the algorithm uses $O(dN + N^{1+\rho})$ space and query time dominated by $O(N^\rho)$ distance computations and $O(N^\rho \log_{1/p_2} N)$ evaluations of hash functions.

2.2.2 LSH for Vectors Based on p-Stable Distributions

Datar, et al. introduced a LSH scheme [DAT04] for $(\mathbb{R}^d, \ell_p)$ based on p-stable distributions. As a reference, a distribution $\mathscr{D}$ is p-stable for some $p \geq 0$ if for any $v_1, \cdots, v_n \in \mathbb{R}$ and i.i.d. variables $X_1, \cdots, X_n \sim \mathscr{D}$, $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where $X \sim \mathscr{D}$ is a r.v. with distribution $\mathscr{D}$. It is well known that a Gaussian distribution is a 2-stable distribution.

A p-stable distribution is very convenient for obtaining a sketch or a hash of a high-dimensional vector by preserving its p-norm. Given a vector $\mathbf{v} \in \mathbb{R}^d$, generate a random vector $\mathbf{a} \sim \mathscr{D}^d$, where $\mathscr{D}$ is a p-stable distribution. Then $\mathbf{a} \cdot \mathbf{v}$ is a r.v. with distribution $||\mathbf{v}||_p X$ where X is a random variable with distribution $\mathscr{D}$ by the definition of a p-stable distribution.

Coming back to the LSH problem, given two points $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{R}^d$, we can use the above technique to get a $\mathbf{a} \cdot (\mathbf{v_1} - \mathbf{v_2})$, which is a r.v. with distribution $||\mathbf{v_1} - \mathbf{v_2}||_p X$ where again X is a random variable with a p-stable distribution. This gives us a good idea to develop hash functions for our LSH scheme. Indyk, et al. defines $h_{\mathbf{a},b}(\mathbf{v}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{r} \rfloor$, where $\mathbf{a} \sim D^d, b \sim U(0, r)$.

If we have $||\mathbf{v_1} - \mathbf{v_2}||_p = c$, and f_p be p.d.f. of the absolute value of the p-stable distribution, then

$$\Pr_{\mathbf{a},b}[h_{\mathbf{a},b}(\mathbf{v_1} = h_{\mathbf{a},b}(\mathbf{v_2})] = \int_0^r \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{r}) dt \tag{1}$$

In this paper, we are most interested about ℓ_2, where we will use the Gaussian distribution as our 2-stable distribution, and for a fixed r and c, we can numerically bound the values of p_1 and p_2, and hence ρ and bounds for analysis of our algorithm.

3 Metric for Frame Distance

3.1 The Frobenius Norm after DCT Transformation

As a reference, for a matrix $A \in \mathbb{R}^{m \times n}$, the Frobenius norm of A is denoted as $||A||_F = (\sum_i^m \sum_j^n ((A)_{i,j})^2)^{1/2}$. The metric d_F induced by the Frobenius norm for two matrices $A, B \in \mathbb{R}^{m \times n}$ is $d_F(A,B) = ||A - B||_F$.

Unfortunately, directly applying this norm onto two matrices representing two images won't work as expected. The distance under this metric for very similar images is on a par with the distance of two dissimilar images. This is because a small displacement or any kinds of variances will cause the images to be misaligned, and the metric induced by the Frobenius norm only works well when the similar components are exactly "aligned".

The matrix under DCT, however, is much more robust to changes. According to previous discussions on DCT, we also have the convenient way to shrink the dimension by omitting higher order coefficients, with out much loss of information. We first define the metric and argue that it is indeed permissible. For $A,B \in R^{n\times n}$

$$d(A,B) := ||DCT(A) - DCT(B)||_F$$

It is clear that $d(A,A) = 0$ and d is symmetric. To show the triangular inequality of d, note that since the d_F satisfies triangular inequality, we have

$$||DCT(A) - DCT(B)||_F \leq ||DCT(A) - C'||_F + ||C' - DCT(B)||_F$$

for all $C' \in R^{d\times d}$. Since the IDCT is well-defined, we can always find a $C = IDCT(C')$, so

$$||DCT(A) - DCT(B)||_F \leq ||DCT(A) - DCT(C)||_F + ||DCT(C) - DCT(B)||_F$$

or $d(A,B) \leq d(A,C) + d(B,C)$ for all $C \in R^{n\times n}$.

3.2 Experiments with the Metric

To test the effectiveness of the metric, we pick five frames from a grayscale video. Three frames ((a)-(c)) are from the same shot, while the other two ((e) and (f)) are from two different shots. We show the distances under d_F, under d, and under d but truncating the matrices after DCT to $\mathbb{R}^{\log n\times\log n}$. The metric d_F is uninformative, and metric d under truncation works best in terms of enlarging the distance gap.

	(a)	(b)	(c)	(d)	(e)
(a)	0	28359	32198	46453	32965
(b)	25522	0	28463	46436	32991
(c)	27568	25951	0	46510	32948
(d)	31646	31688	31626	0	33600
(e)	44235	44202	44228	41617	0

Table 1: Distances under d_F

	(a)	(b)	(c)	(d)	(e)
(a)	0	1752	2348	14137	16643
(b)	1752	0	2197	14125	16659
(c)	2348	2197	0	14113	16648
(d)	14137	14125	14113	0	20800
(e)	16643	16659	16648	20800	0

Table 2: Distances under d

	(a)	(b)	(c)	(d)	(e)
(a)	0	370	509	12225	15562
(b)	370	0	409	12210	15601
(c)	509	409	0	12145	15586
(d)	12225	12210	12145	0	19627
(e)	15562	15601	15586	19627	0

Table 3: Distances under d with truncation

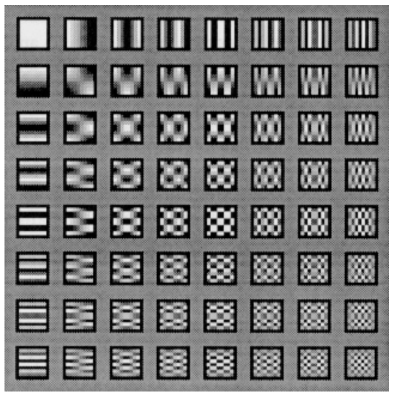
Figure 1

Figure 2(a)

Figure 2(b)

Figure 2(c)

Figure 2(d)

Figure 2(e)

Figure 1. Visualizations of the basis of the 2-d DCT basis ordered in increasing frequencies. Figure 2. Sample frames. 2(a)-(c) are similar frames from the same shot. 2(d) and (e) are from two different shots.

4 Algorithm for Key Frame Detection

```
Algorithm 1: Key Frame Detection
Input: $r, c$ approximation parameters, $N$ the number of frames, $n$ the size of a frame;
Calculate $p_1(r,c), p_2(r,c)$ based on Eqn 1 for Normal distribution, $k := \log_{1/p_2} N$, $L := N^{p_2/p_1}$;
Initialize a hash-map $m$ that maps from $\mathbb{R}^k$ to a set of vectors in $\mathbb{R}^{\log^2 n}$, initialize an empty set S;
for each frame $F$ do
    1. $D = DCT(F)$, truncate $D$ to $D' \in \mathbb{R}^{\log n \times \log n}$ and linearize D' to $\mathbf{v} \in \mathbb{R}^{\log^2 n}$;
    2. Generate $L$ hashes $g_{i\in[L]}(\mathbf{v}) = (h_{i_1}, \cdots, h_{i_k})$, where $h_{i_{j\in[k]}} = \lfloor \frac{\mathbf{a}\cdot\mathbf{v}+b}{r} \rfloor$, $\mathbf{a} \sim \mathcal{N}(0,1)^{\log^2 n}$, and
       $b \sim U(0,r)$. Insert $\mathbf{v}$ to buckets corresponding to the hashes;
    3. Check up to $3L$ vectors in the buckets corresponding to $g_{i\in[L]}$, if $\ell_2(u,v) > cr$ for each
       vector $u$ from the $3L$ vectors, insert $F$ to $S$.
end
return F;
```

With the previously defined metric d, we can formulate the distance between two frames as the

Frobenius norm after DCT (and truncation for space saving). This enables us to apply the LSH scheme for vectors under ℓ_2. The full algorithm is presented at Algorithm 1.

The algorithm uses $O(\log^2(n)N + N^{1+\rho})$ space and query time dominated by $O(N^\rho)$ distance computations and $O(N^\rho \log_{1/p_2} N)$ evaluations of hash functions.

We also tested the algorithm on a sample 5-minute gray-scale video with 8439 frames. The unoptimized python implementation of the algorithm takes 27.49 seconds of real time computation to extract 64 frames. The video compiled with the 64 frames provides a relatively good summary of the original video, but it also showcases several problems:

1. The key frames captured are almost always the first frame of a shot, which may not be the best representation of a shot. The quality of the frame is also rather blurry. We may want to add additional mechanisms to select a best representation frame among several neighbors.
2. Several similar frames are repeated while others are not captured, which is mostly likely due to the algorithm's probabilistic failure and imperfect hyper-parameters.
3. Several key frames to the end are not captured, because they are similar to frames at the beginning.

5 Conclusions

We propose to formulate the problem of detecting key frames of a video under the streaming setting as a locality-sensitive hashing (LSH) for matrices. For this end, we define a metric for matrices using discrete cosine transform (DCT) and the Frobenius norm, which works practically for video frames. We then adapt the LSH scheme for ℓ_2 and show some qualitative experimental results. We observe several flaws of the algorithm based on the experiment as mentioned in the previous section, and future work may focus on improving on these problems. We may also focus on finding or devising an appropriate quantitative metric for key frame detection task to evaluate and improve the performance of the algorithm.

6 References

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, c-23(1): 90-93, Jan. 1974.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the thirtieth annual ACM Symposium on Theory of computing*: 604-613, May 1998.

[Dat04] Mayur Datar, et al. Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the twentieth annual symposium on Computational geometry*: 253-262, June 2004.