

Multimedia Information Retrieval



Prof Stefan Ruger
Multimedia and Information Systems
Knowledge Media Institute
The Open University
<http://kmi.open.ac.uk/mmis>

MMIS

Multimedia and Information Systems



1. What are multimedia queries?
2. Fingerprinting
3. Metadata & piggy-back retrieval
4. Automated image annotation
- 5 Visual content-based retrieval I
- 6 Visual content-based retrieval II
7. Evaluation
8. Browsing, search and geography



1. What are multimedia queries?
2. Fingerprinting
 - Overview
 - How Shazam works
 - Subfingerprinting
 - Locality sensitive hashing
 - Min hash algorithm
 - Sift image features
 - Surf as alternative
3. Metadata & piggy-back retrieval
4. Automated image annotation
5. Visual content-based retrieval I
6. Visual content-based retrieval II
7. Evaluation
8. Browsing, search and geography



Fingerprinting technique

- 1 Compute salient points
- 2 Extract “characteristics” from vicinity (feature)
- 3 Make invariant under rotation & scaling
- 4 Quantise: create visterms
- 5 Index as in text search engines
- 6 Check/enforce spatial constraints after retrieval



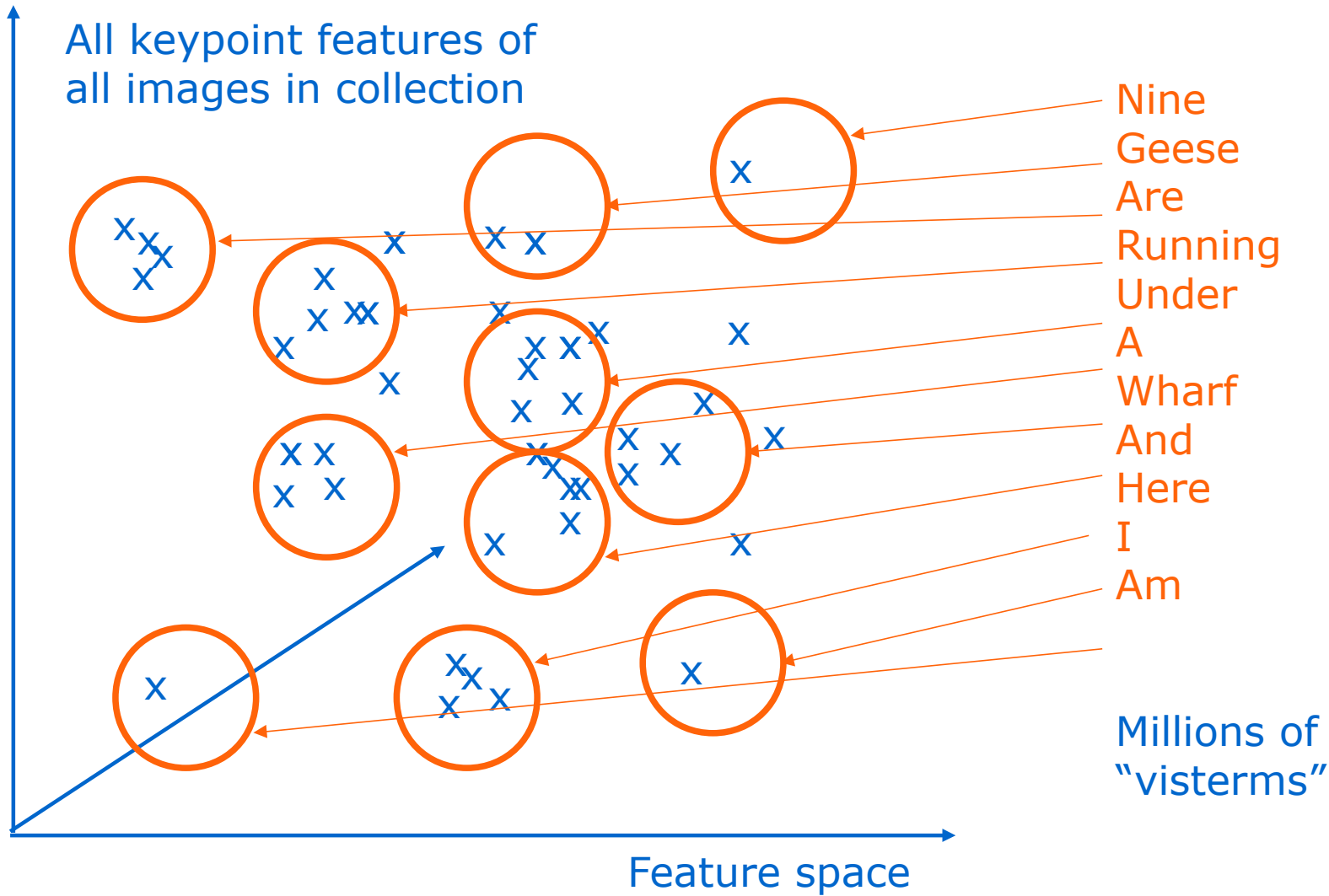
NDD: Compute salient points and features



Eg, SIFT features: each salient point described by a feature vector of 128 numbers; the vector is invariant to scaling and rotation



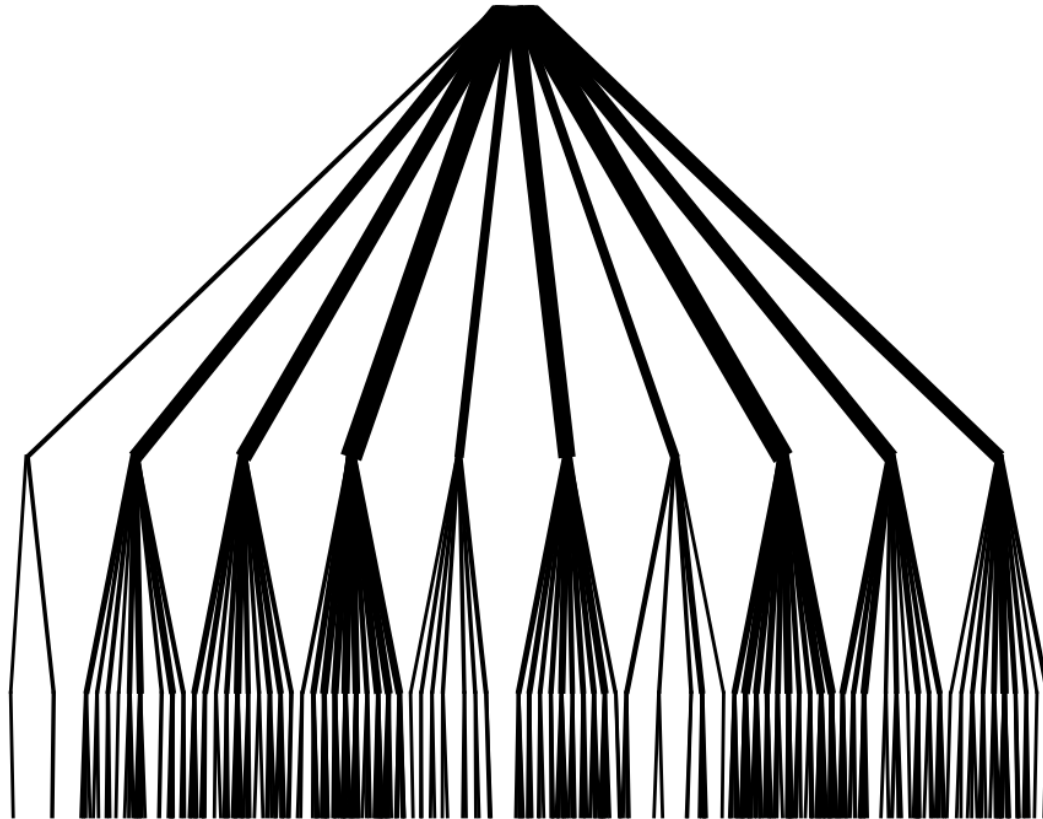
NDD: Keypoint feature space clustering





Clustering

Hierarchical k-means





NDD: Encode all images with visterms



Jkjh Geese Bjlkj Wharf
Ojkkjhjhj Kssn Klkekjl Here
Lkjkll Wjjkll Kkjlk Bnm
Kllkgjg Lwoe Boerm ...



At query time compute salient points,
keypoint features and visterms

Query against database of images
represented as bag of visterms

Query



Joiu Gddwd Bipoi Wueft
Oiooiuui Kwwn Kpodoip Hdfd
Loiopp Wiiopp Koipo Bnm
Kppoyiy Lsld Bldfm ...



NDD: Check spatial constraints

Learning in an Open World
Open University: Online Conference, June 2010
<http://cloudscapes.org.uk/cloudscapes/v/1aw/2012>

SocialLearn Team: Simon Buckingham Shum,
Rebecca Ferguson, Mark Glaister, Thanh Le

<http://creativecommons.org/licenses/by-nc-sa/4.0/>



Fingerprinting technique

- 1 Compute salient points
- 2 Extract “characteristics” from vicinity (feature)
- 3 Make invariant under rotation & scaling
- 4 Quantise: create visterms
- 5 Index as in text search engines
- 6 Check/enforce spatial constraints after retrieval



How Shazam works - Spectrogram

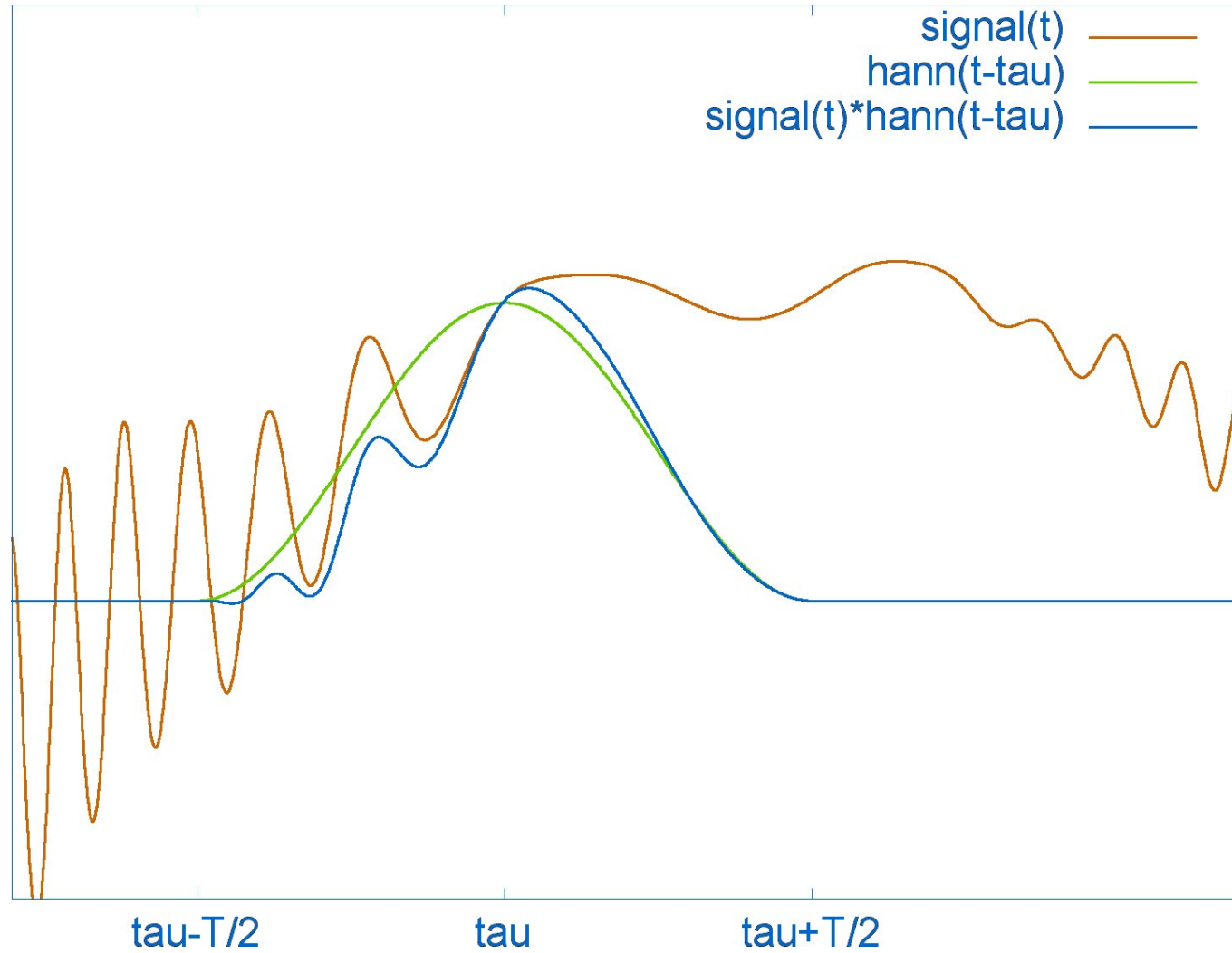
Compute energy for all (frequency,time) pairs
 using a Fourier transform under a Hann window w

$$\text{spectrogram}(f, \tau) = \left| \int_{-\infty}^{\infty} s(t)w(t - \tau)e^{jft} dt \right|^2$$

$$w(t) = \begin{cases} \frac{1}{2} + \frac{1}{2} \cos\left(\frac{2\pi t}{T}\right) & \text{if } t \in [-T/2, T/2] \\ 0 & \text{otherwise} \end{cases}$$

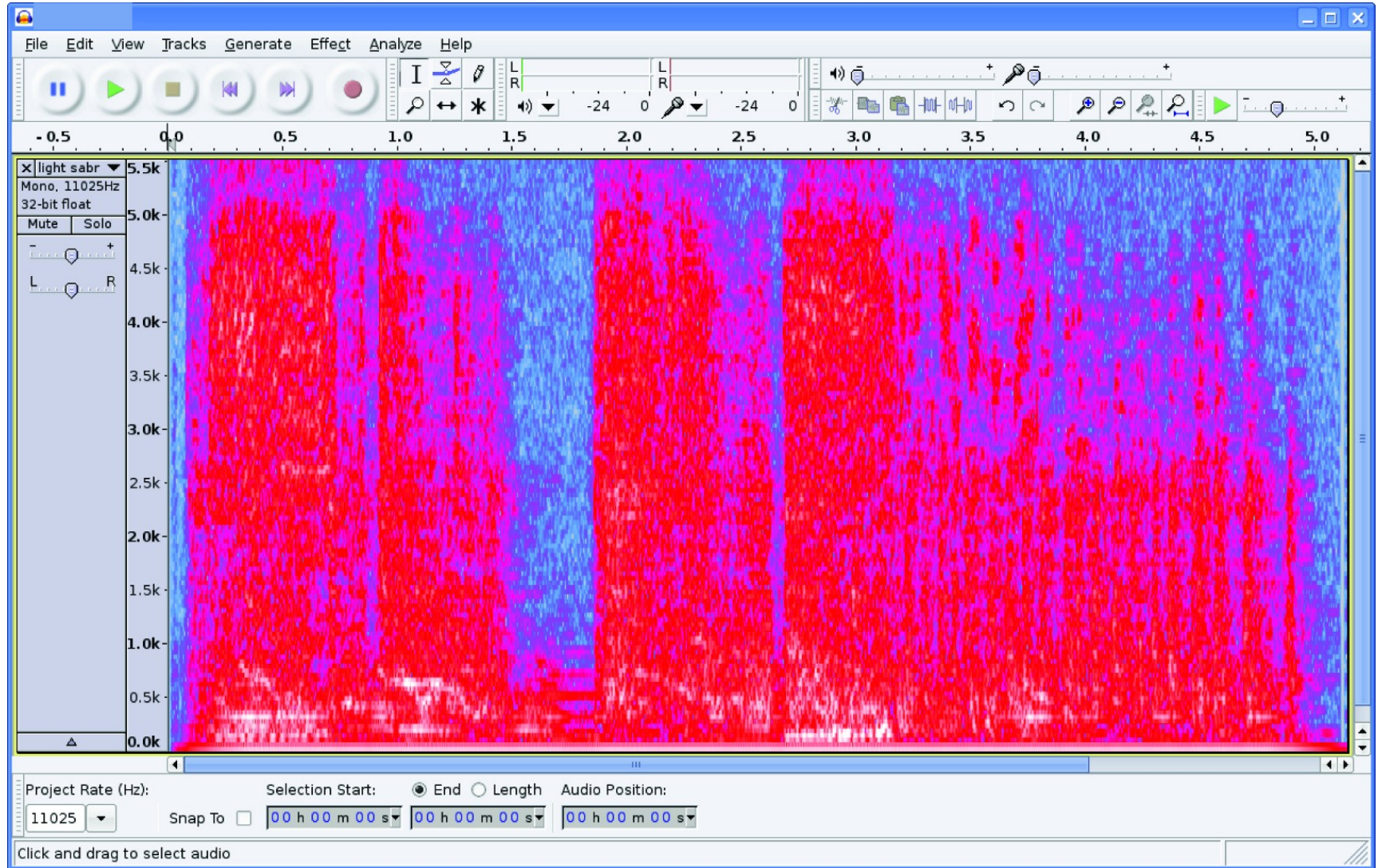


Hann window application



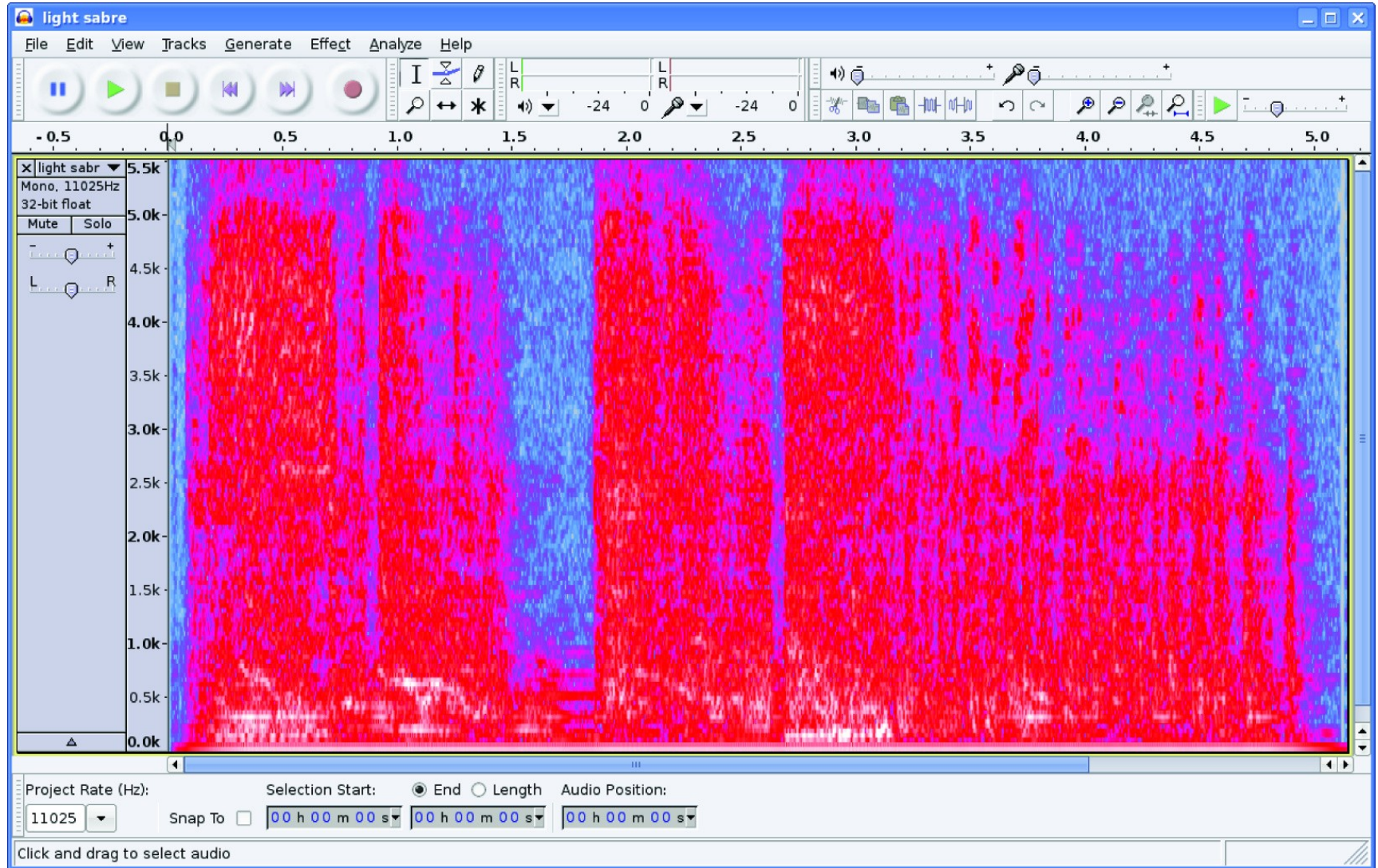


How Shazam works: audio fingerprinting



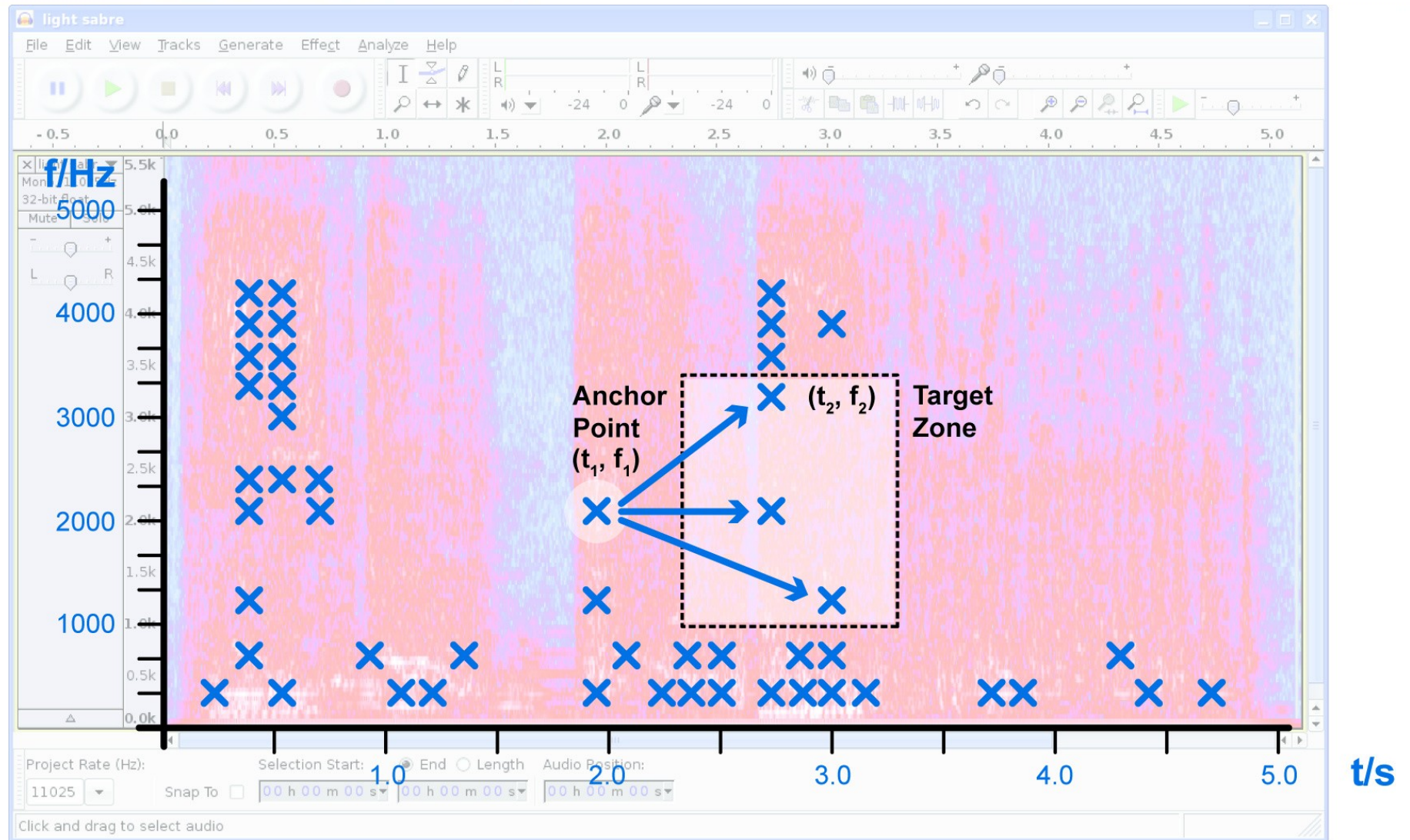


How Shazam works: audio fingerprinting





Salient points



Encoding: $(f_1, f_2, t_2 - t_1)$ hashes to (t_1, id)

[Wang(2003), An industrial-strength search algorithm, ISMIR]



Temporal consistency check of query

Every query vector $(f_1, f_2, t_2^q - t_1^q)$ is matched to the database.

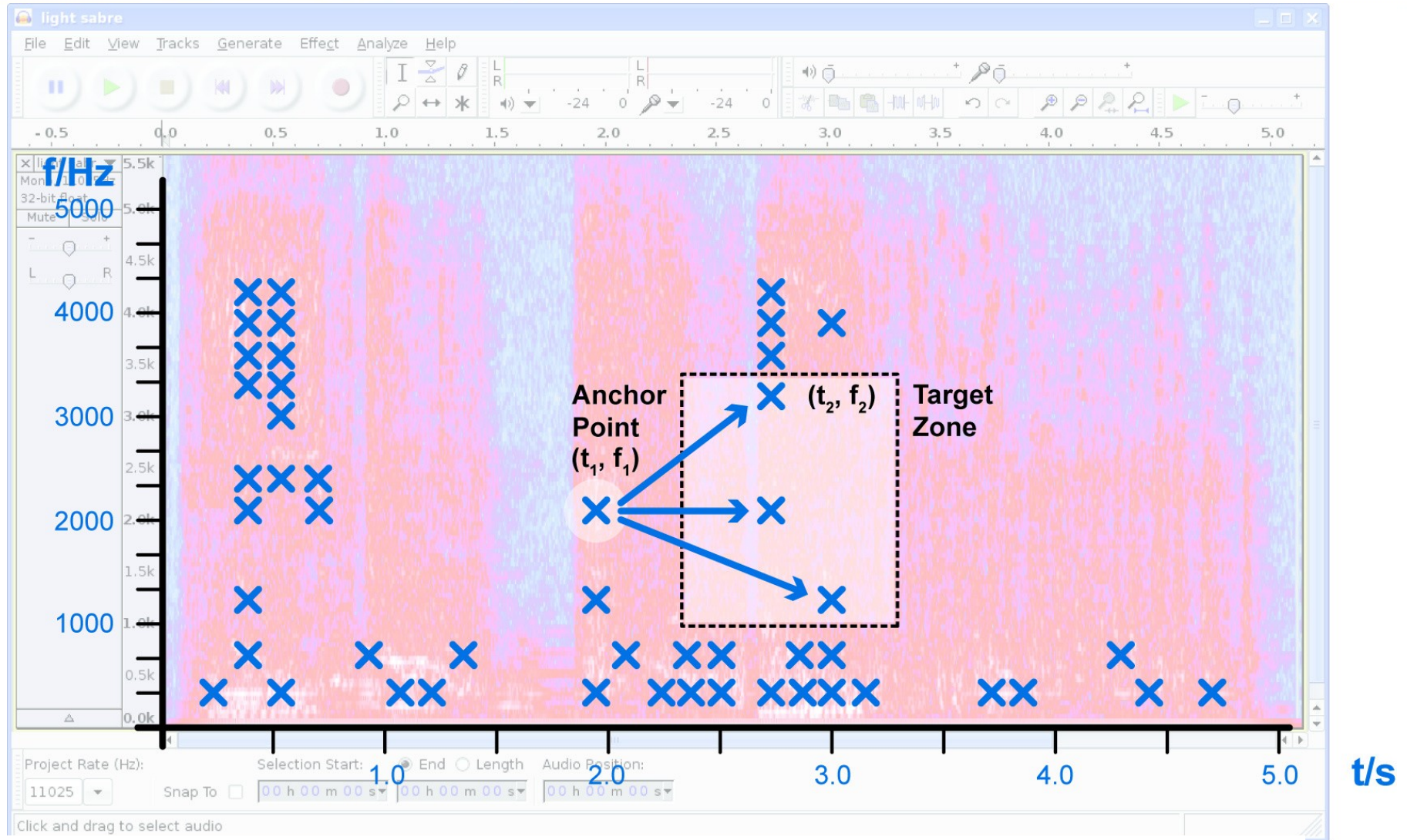
You get a list of possible (t_1^{id}, id) values (some are false positives).

Create a histogram of $t_1^{id} - t_1^q$ values (temporal consistency check!)

A substantial peak in this histogram means that the query has matched song id at time offset $t_1^{id} - t_1^q$.



Entropy considerations



Specificity: Encoding $(f_1, f_2, t_2 - t_1)$ to use 30 bit



Exercise Shazam's constellation pairs

Assume that the typical survival probability of each 30-bit constellation pair after deformations that we still want to recognise is p , and that this process is independent per pair. Which encoding density, ie, the number of constellation pairs per second, would you need on average so that a typical query of 10 seconds exhibits at least 10 matches in the right song with a probability of at least 99.99%? Under these assumptions, further assuming that the constellation pair extraction looks like a random independent and identically distributed number, what is the false positive rate for a database of 4 million songs each of which is 5 minutes long on average?



Divide frequency scale into 33 frequency bands between 300 Hz and 2000 Hz
Logarithmic spread – each frequency step is 1/12 octave, ie, one semitone

Divide time axis into blocks of 256 windows of 11.6 ms (3 seconds)

$E(m,n)$ is the energy of the m -th frequency at n -th time in spectrogram

For each block extract 256 sub-fingerprints of 32 bits each

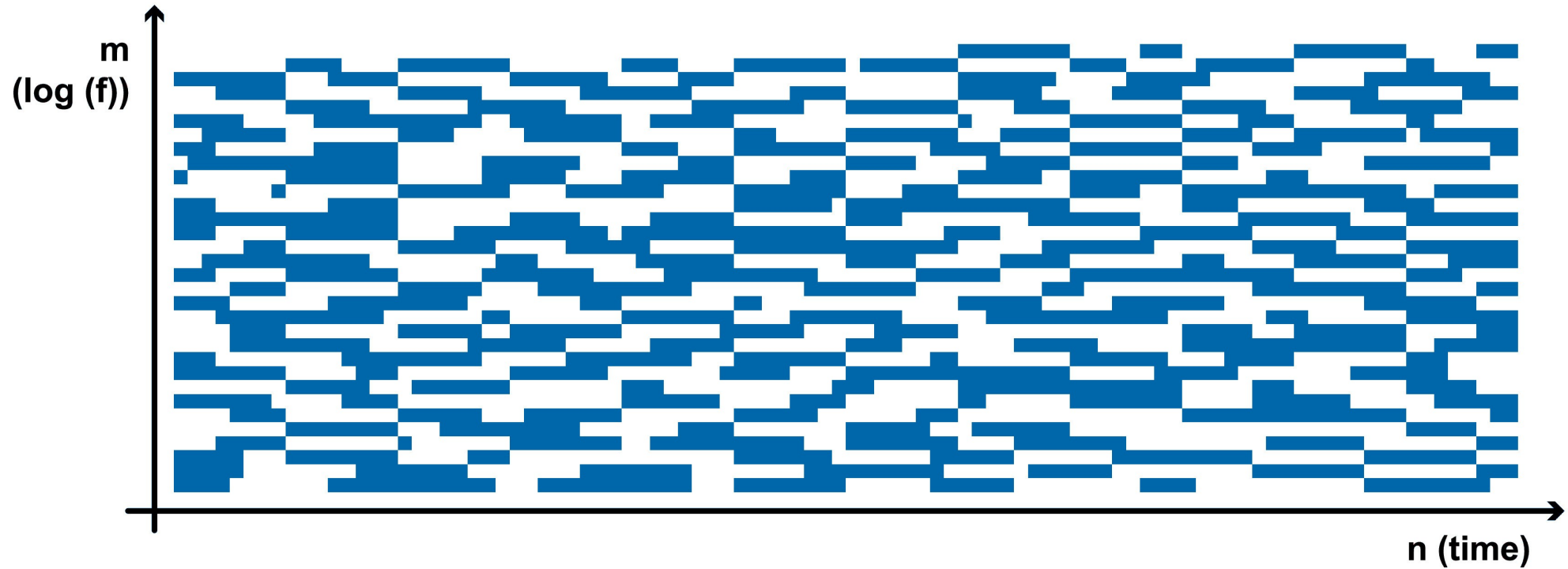
$$b(m, n) = \text{sign} ([E(m, n) - E(m + 1, n)] - [E(m, n + 1) - E(m + 1, n + 1)])$$

$$0 \leq m \leq 31 \text{ (frequency)}$$

$$0 \leq n \leq 255 \text{ (time)}$$

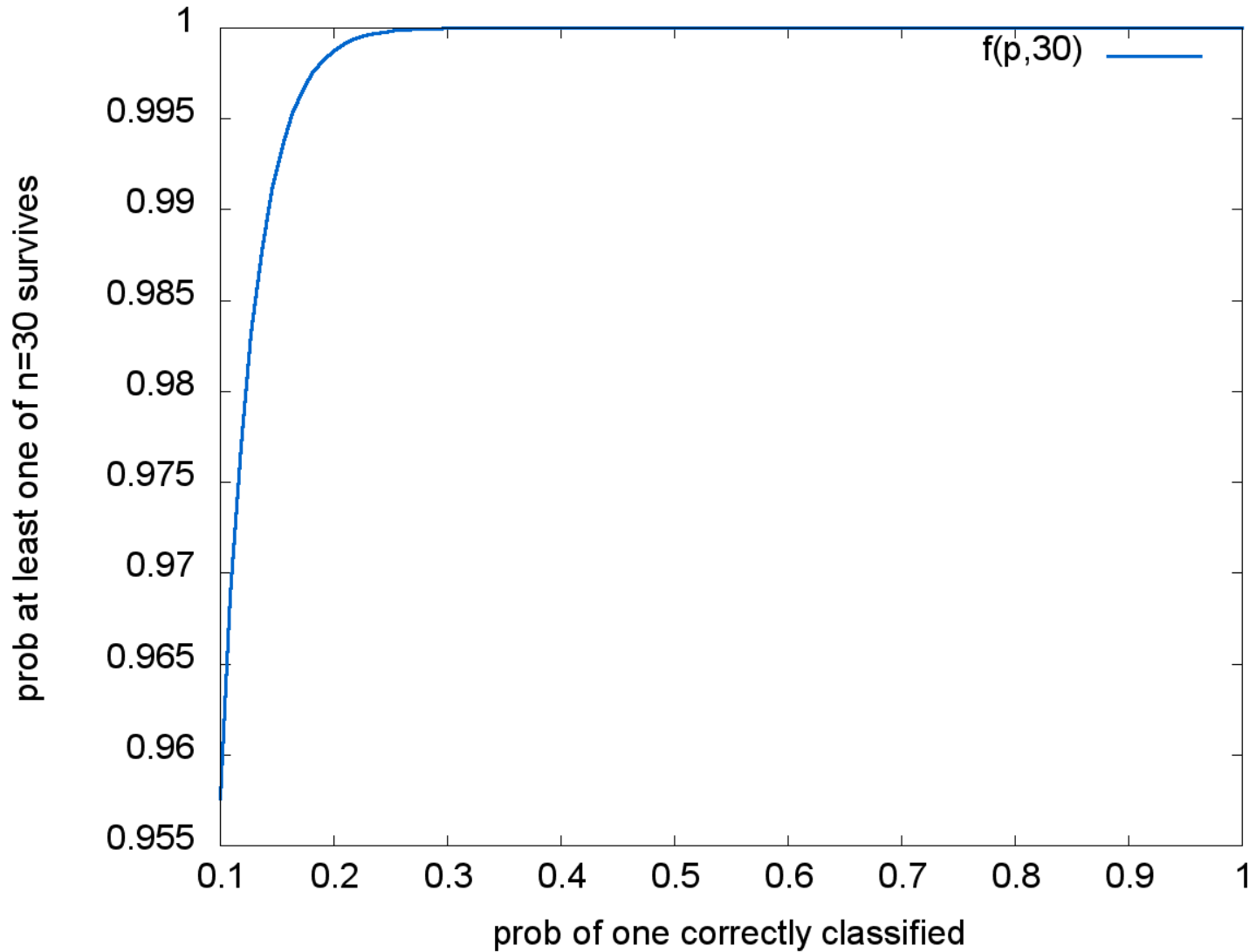


Partial fingerprint block





Redundancy Probability of matching





Exercise: fingerprint block probabilities

Assuming bit errors are independently identically distributed at the rate of b . Show that the probability $p(k, b)$ of having no more than k bit errors in one sub-fingerprint is

$$p(k, b) = \sum_{i=0}^k \binom{32}{i} (1 - b)^{32-i} b^i.$$

Show that the probability that among 256 sub-fingerprints at least one survives with no more than k bit errors is given by

$$1 - (1 - p(k, b))^{256}.$$

Verify, using above formulas, the following claim: Even though a high bit error rate of $b = 0.3$ causes the probability $p(4)$ that no more than 4 bits were flipped to drop under 2%, it is the case that when you look at 256 sub-fingerprints, at least one of them will have no more than 4 bit errors with more than 99% probability.

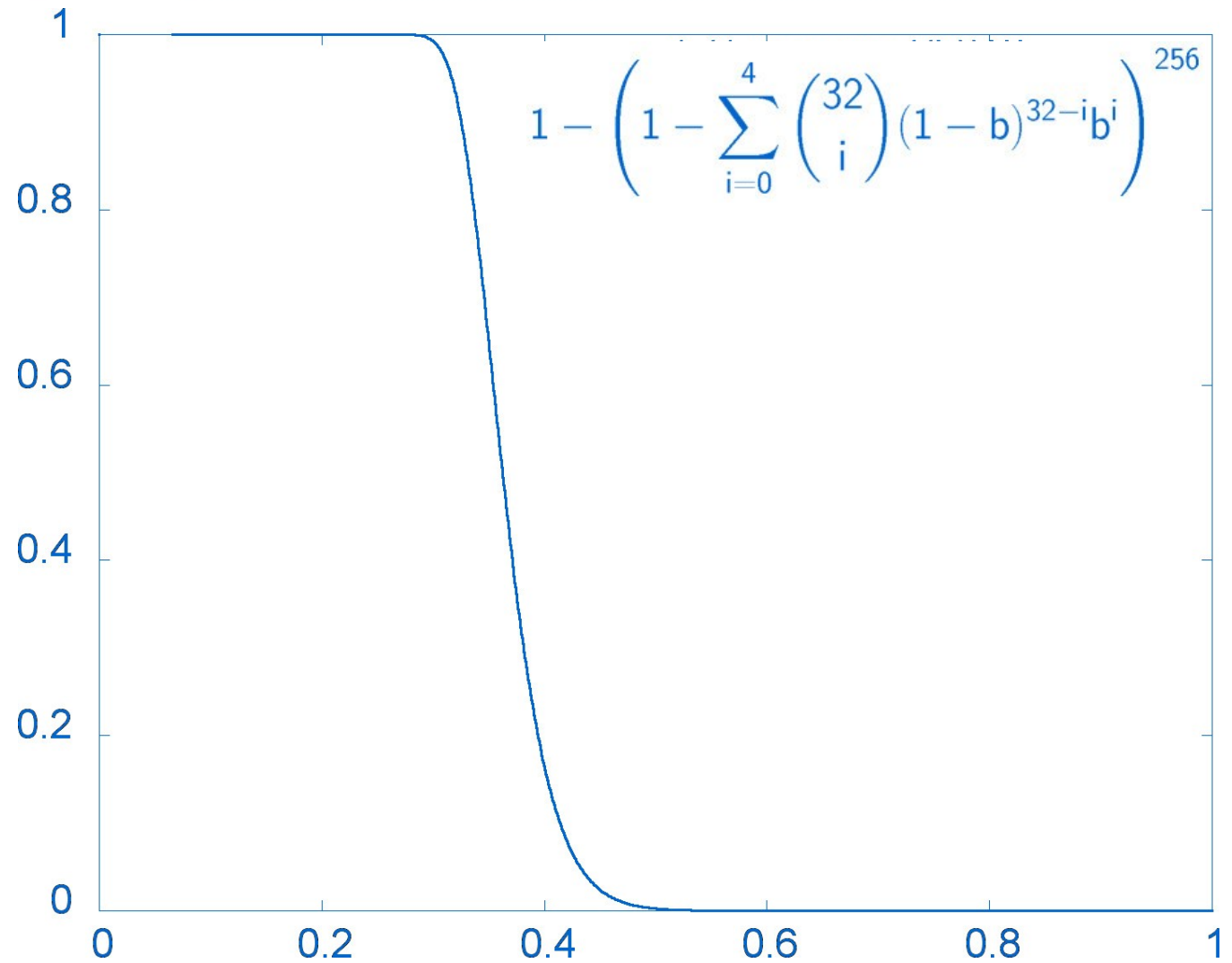


Exercise: fingerprint block false positives

Assuming that the fingerprint block extraction process yields random, independent and identically distributed bits, what is the probability that a randomly modified fingerprint block matches a *different* random block in the database that consists of, say, 10^{11} overlapping fingerprint blocks (4 million songs with around 5 minutes each)? The bit error rate for the random modification is assumed to be 35%.



Probability of at least one sub-fingerprint surviving with no more than 4 errors





How audio matching with sub-fingerprinting works



Quantisation through locality sensitive hashing (LSH)

$$h^i: \mathbb{R}^d \rightarrow \mathbb{Z}$$

$$\mathbf{v} \mapsto h^i(\mathbf{v}) = \left\lfloor \frac{\mathbf{a}^i \mathbf{v} + b^i}{w} \right\rfloor$$

$\mathbf{a}^i \in \mathbb{R}^d$ is a random Gaussian-distributed vector

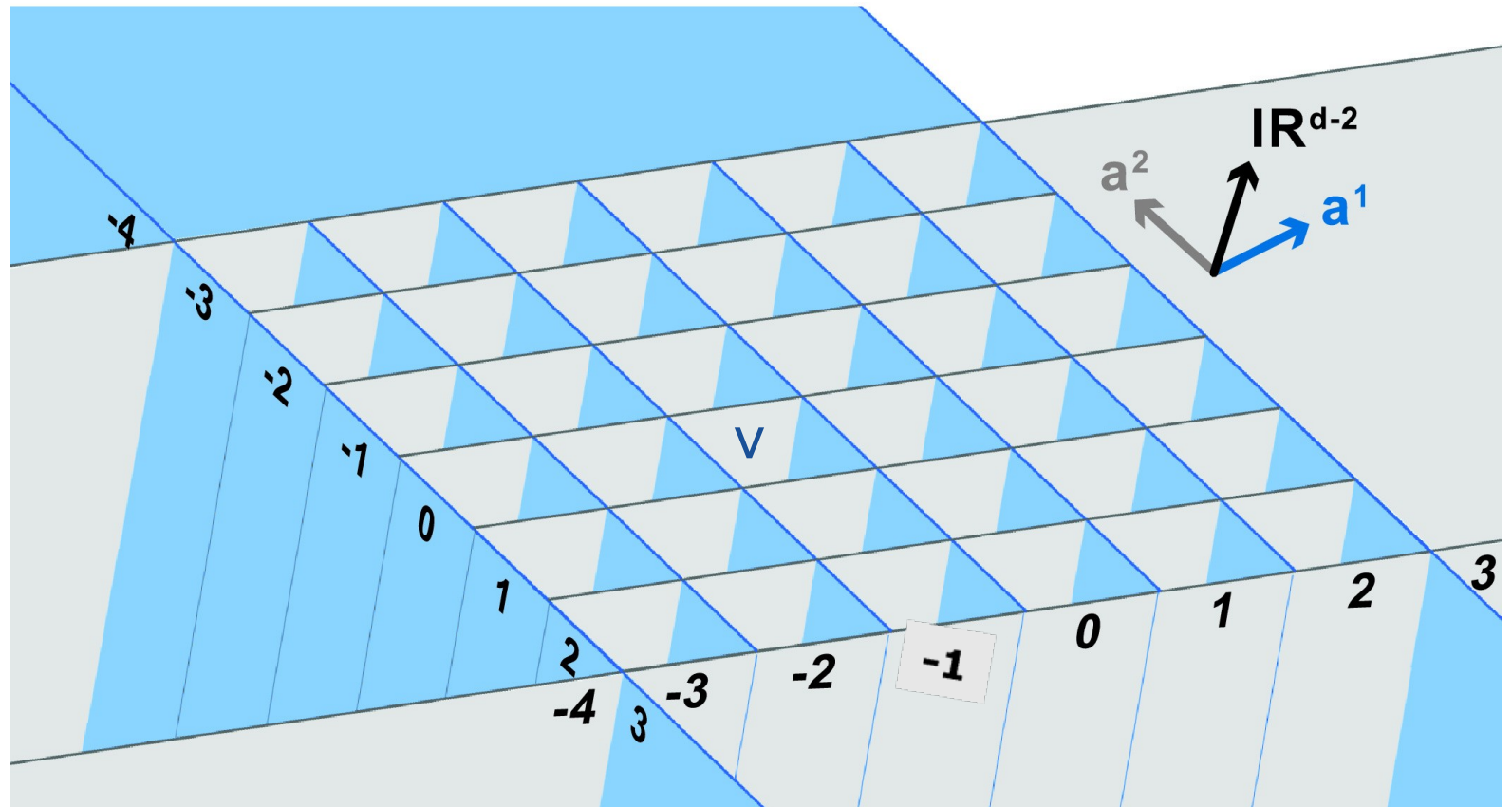
$w \in \mathbb{R}^+$ is a constant

$b^i \in [0, w)$ is a random number

$\mathbf{h}(\mathbf{v}) = (h^1(\mathbf{v}), h^2(\mathbf{v}), \dots, h^k(\mathbf{v}))$ is the LSH hash vector.



Quantisation LSH hashes



Vector v : $h(v) = (-1, 0)$



Redundancy is key

Use L independent hash vectors of k components each both for the query and for each multimedia object.

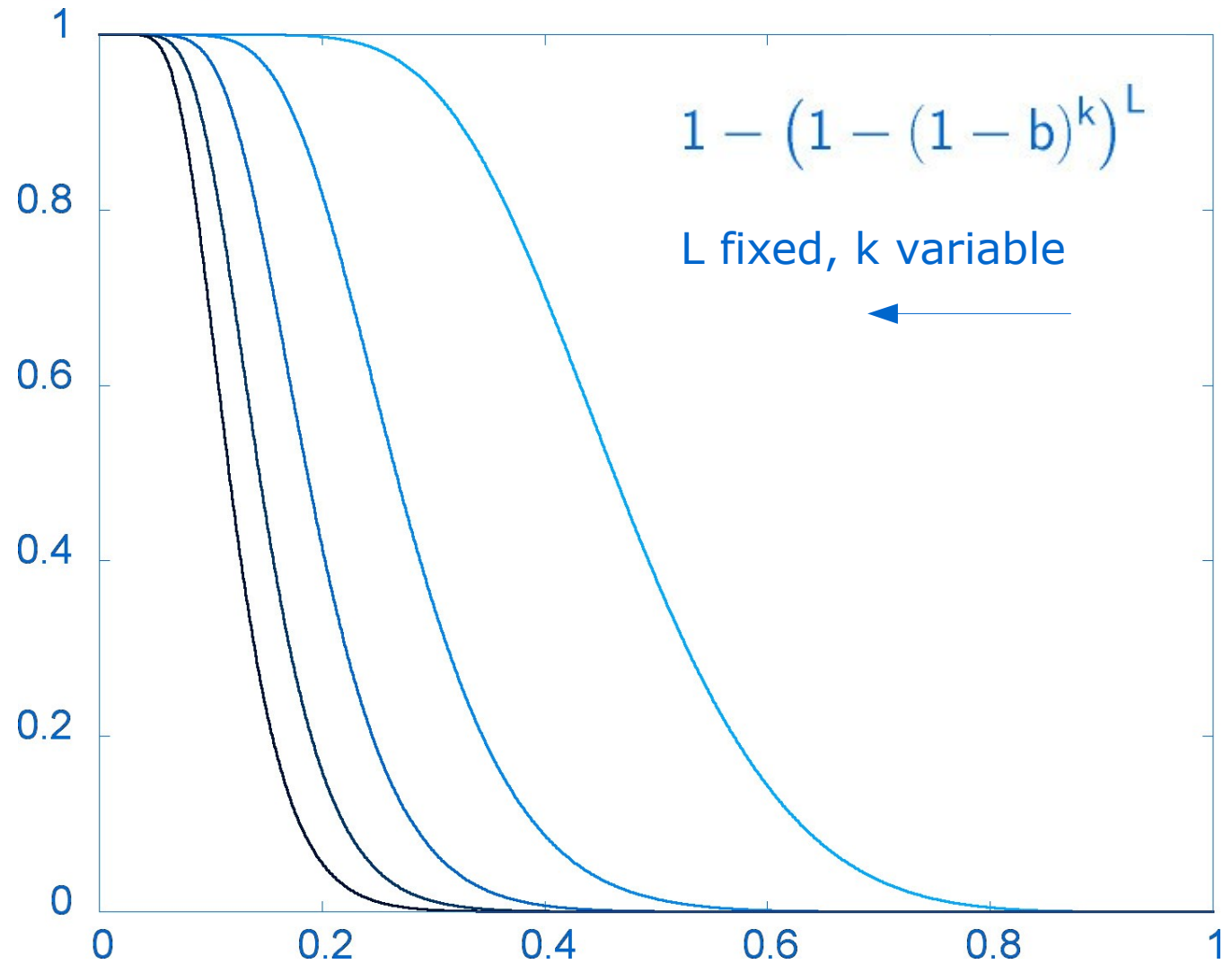
Database elements that match at least m out of L times are candidates for nearest neighbours.

Chose w , k and L (wisely) at runtime

- w determines granularity of bins, ie, # of bits for $h^i(v)$
- k and L determine probability of matching

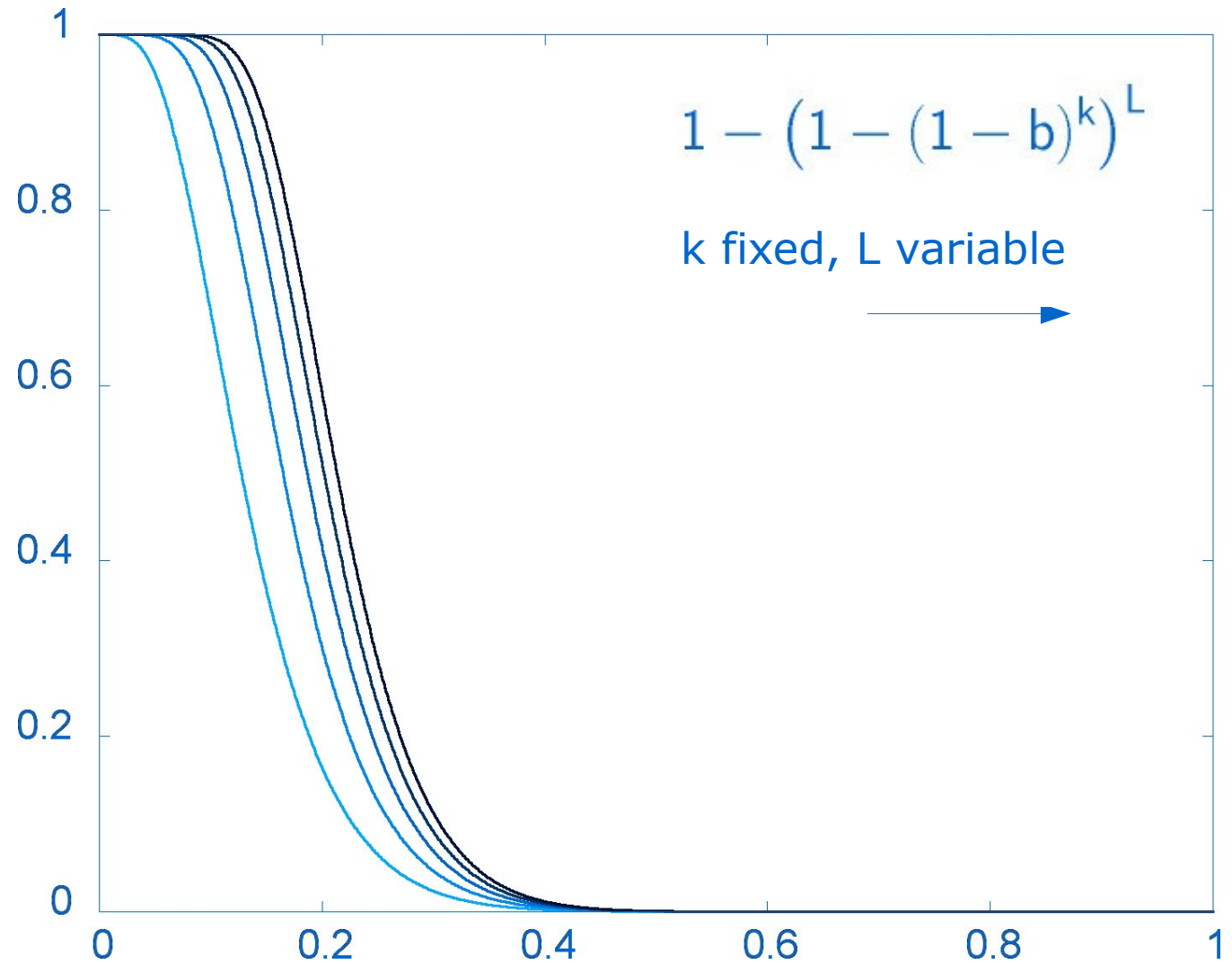


Prob(min 1 match out of L)



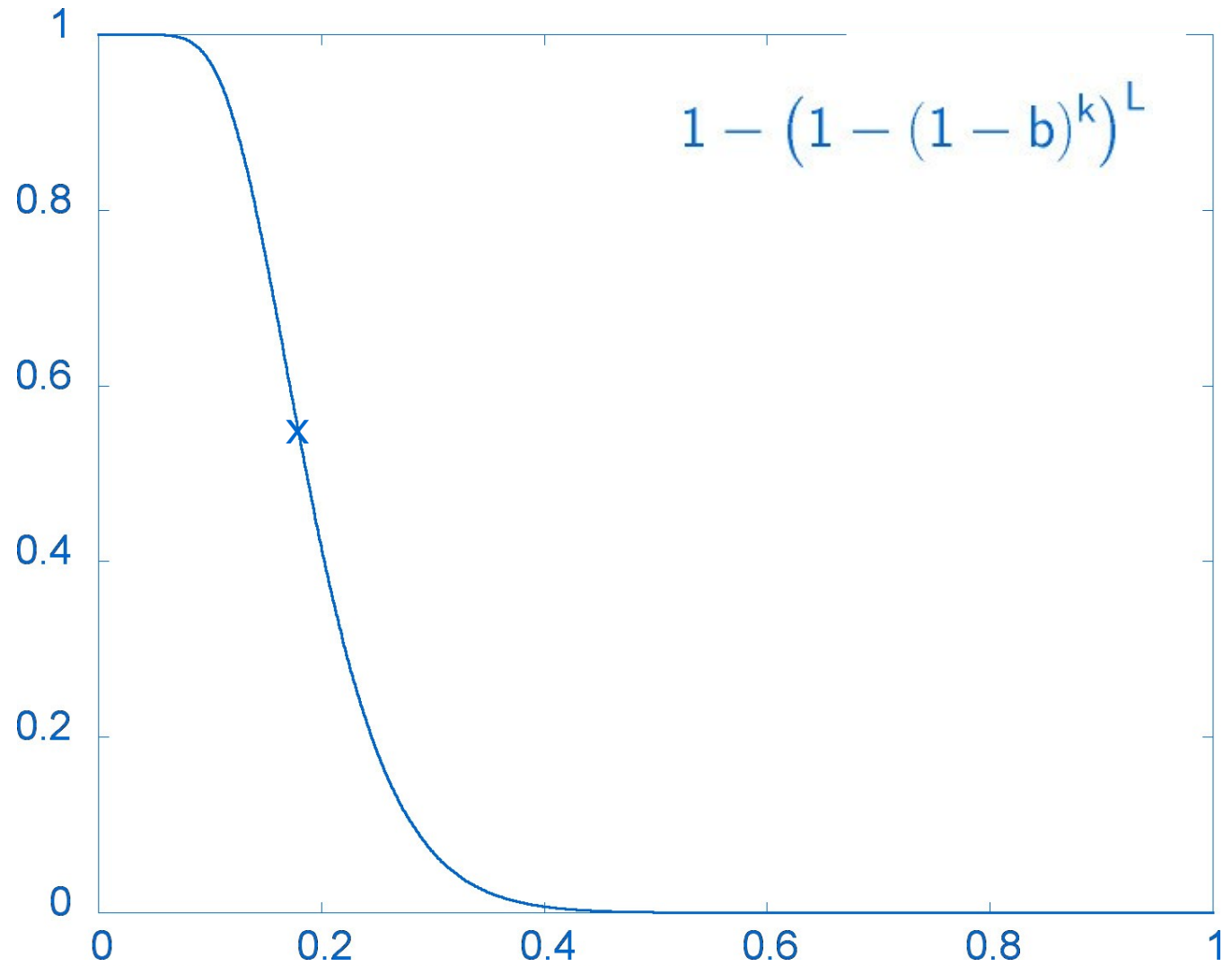


Prob(min 1 match out of L)





Exercise: compute inflection point





Min hash

Estimate discrete set overlap

$$\text{sim}(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}$$



An example 4 documents

D_1 = Humpty Dumpty sat on a wall,

D_2 = Humpty Dumpty had a great fall.

D_3 = All the King's horses, And all the King's men

D_4 = Couldn't put Humpty together again!



Surrogate docs after stop word removal and stemming

$A_1 = \{\text{humpty, dumpty, sat, wall}\}$

$A_2 = \{\text{humpty, dumpty, great, fall}\}$

$A_3 = \{\text{all, king, horse, men}\}$

$A_4 = \{\text{put, humpty, together, again}\}$



Equivalent term-document matrix

	A_1	A_2	A_3	A_4
humpty	1	1	0	1
dumpty	1	1	0	0
sat	1	0	0	0
wall	1	0	0	0
great	0	1	0	0
fall	0	1	0	0
all	0	0	1	0
king	0	0	1	0
horse	0	0	1	0
men	0	0	1	0
put	0	0	0	1
together	0	0	0	1
again	0	0	0	1



Similarity between two docs

$$\text{sim}(A_i, A_j) = \frac{c_{11}}{c_{11} + c_{10} + c_{01}}$$

c_{xy} = number of (x,y) rows

Important observation
 c_{00} is unused!

	A_1	A_2	A_3	A_4
humpty	1	1	0	1
dumpty	1	1	0	0
sat	1	0	0	0
wall	1	0	0	0
great	0	1	0	0
fall	0	1	0	0
all	0	0	1	0
king	0	0	1	0
horse	0	0	1	0
men	0	0	1	0
put	0	0	0	1
together	0	0	0	1
again	0	0	0	1



Estimation of similarity through random permutations

$\pi_1 =$ (dumpty, men, again, put, great, humpty, wall, horse, king, sat, fall, together, all)

$\pi_2 =$ (fall, put, all, again, dumpty, sat, men, great, wall, king, horse, humpty, together)

$\pi_3 =$ (horse, dumpty, wall, humpty, great, again, sat, all, men, together, put, king, fall)

$\pi_4 =$ (king, humpty, men, together, great, fall, horse, all, dumpty, wall, sat, again, put)



Surrogate documents form random permutations

Keep first occurring word of A_i in π_j for dense surrogate representation

	A_1	A_2	A_3	A_4
π_1	dumpty	dumpty	men	again
π_2	dumpty	fall	all	put
π_3	dumpty	dumpty	horse	humpty
π_4	humpty	humpty	king	humpty



Surrogate documents form random permutations

	A_1	A_2	A_3	A_4
π_1	dumpty	dumpty	men	again
π_2	dumpty	fall	all	put
π_3	dumpty	dumpty	horse	humpty
π_4	humpty	humpty	king	humpty

Estimate $\text{sim}(A_2, A_4) = 1/4$
 (proportion of co-occurring words)



Exercise: estimate $\text{sim}(A_4, A_5)$ with min hash

Mice are dancing in a round,
On a bench a cat is sleeping.
"Hush, you mice, don't make such noise
Or you'll wake up Vaska Cat
Vaska Cat will jump and leap
And will spoil and break your round".

$$A_1 = \{\text{mice, danc, round}\}$$

$$A_2 = \{\text{bench, cat, sleep}\}$$

$$A_3 = \{\text{hush, mice, nois}\}$$

$$A_4 = \{\text{wake, vaska, cat}\}$$

$$A_5 = \{\text{vaska, cat, jump, leap}\}$$

$$A_6 = \{\text{spoil, break, round}\}$$



Exercise: estimate $\text{sim}(A_4, A_5)$ with min hash

$A_4 = \{\text{wake, vaska, cat}\}$

$A_5 = \{\text{vaska, cat, jump, leap}\}$

$\pi_1 = (\text{bench, break, cat, danc, hush, jump, leap, mice, nois, round, sleep, spoil, vaska, wake})$

$\pi_2 = (\text{cat, vaska, wake, bench, danc, nois, leap, jump, sleep, round, mice, break, spoil, hush})$

$\pi_3 = (\text{hush, break, vaska, nois, jump, mice, sleep, spoil, wake, round, leap, bench, danc, cat})$

$\pi_4 = (\text{spoil, bench, cat, wake, nois, leap, danc, sleep, jump, round, mice, hush, break, vaska})$

$\pi_5 = (\text{vaska, danc, leap, break, round, nois, spoil, hush, wake, jump, sleep, cat, bench, mice})$

$\pi_6 = (\text{round, vaska, danc, wake, spoil, hush, sleep, leap, cat, nois, mice, break, bench, jump})$

$\pi_7 = (\text{danc, break, vaska, spoil, sleep, wake, round, bench, leap, hush, jump, mice, nois, cat})$

$\pi_8 = (\text{jump, nois, break, danc, round, leap, hush, cat, sleep, vaska, spoil, mice, bench, wake})$



Scale Invariant Feature Transform

“distinctive invariant image features that can be used to perform reliable matching between different views of an object or scene.”

Invariant to image scale and rotation.

Robust to substantial range of affine distortion, changes in 3D viewpoint, addition of noise and change in illumination.

[Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60, 2, pp. 91-110.]



For a given image:

Detect scale space extrema

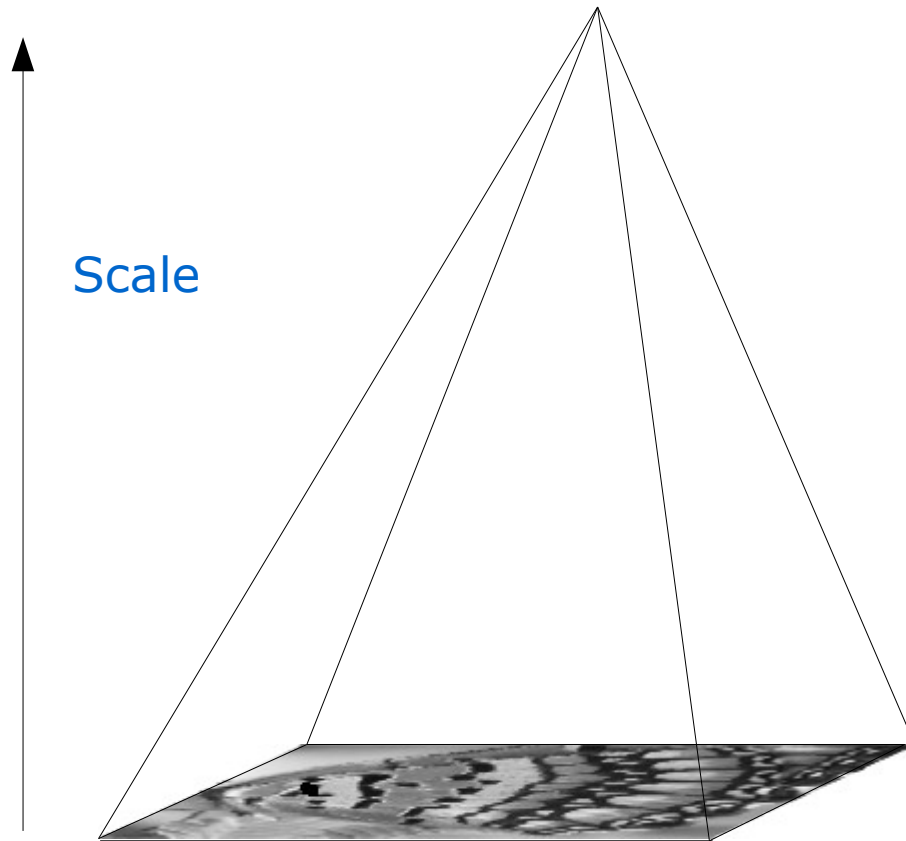
Localise candidate keypoints

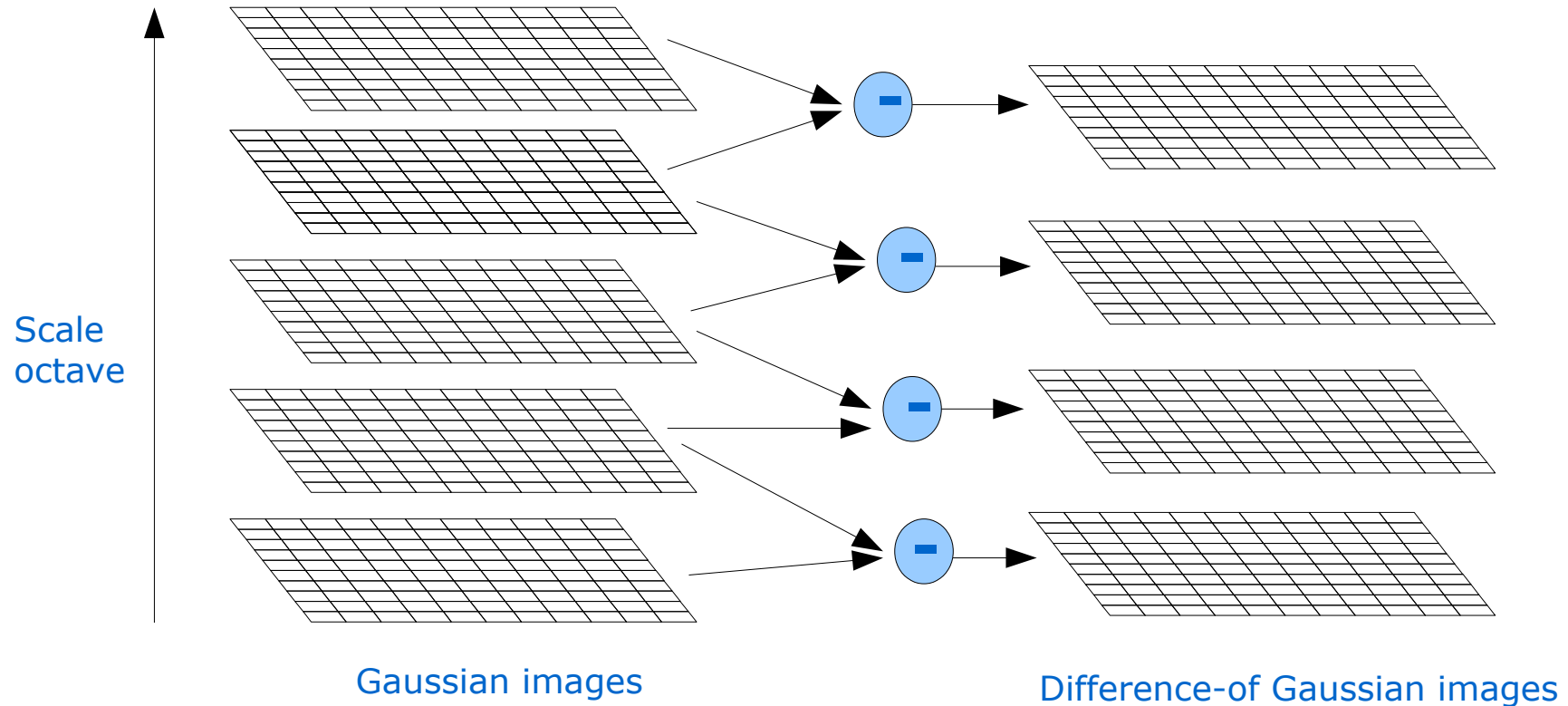
Assign an orientation to each keypoint

Produce keypoint descriptor



A scale space visualisation







Gaussian blur illustration





Difference of Gaussian illustration



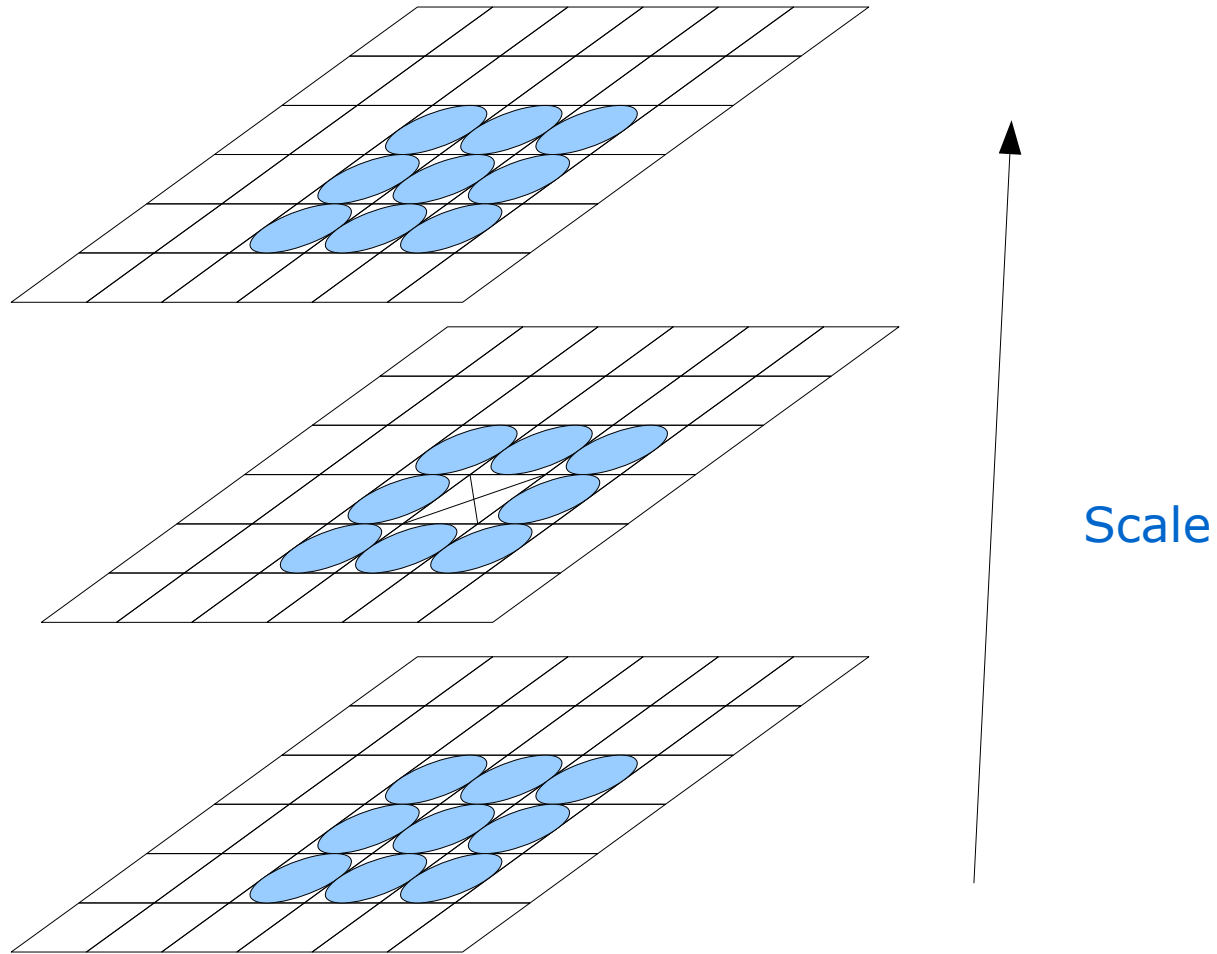


Once the Difference of Gaussian images have been generated:

- Each pixel in the images is compared to 8 neighbours at same scale.
- Also compared to 9 corresponding neighbours in scale above and 9 corresponding neighbours in the scale below.
- Each pixel is compared to 26 neighbouring pixels in 3x3 regions across scales, as it is not compared to itself at the current scale.
- A pixel is selected as a SIFT keypoint only either if its intensity value is extreme.

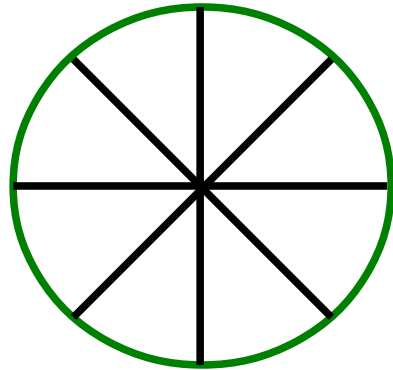


Pixel neighbourhood comparison





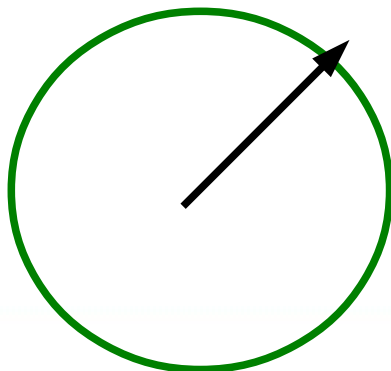
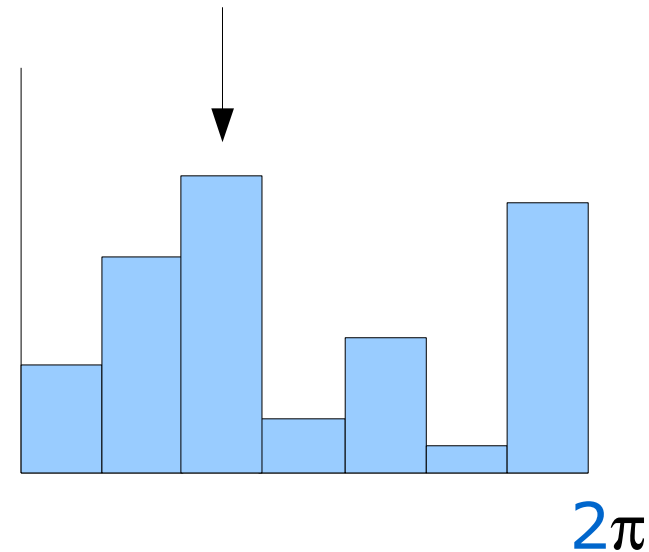
Orientation assignment



Orientation histogram with 36 bins – one per 10 degrees.

Each sample weighted by gradient magnitude and Gaussian window.

Canonical orientation at peak of Smoothed histogram.



Where two or more orientations are detected keypoints created for each orientation.



We now have location, scale and orientation for each SIFT keypoint (“keypoint frame”).

→ descriptor for local image region is required.

Must be as invariant as possible to changes in illumination and 3D viewpoint.

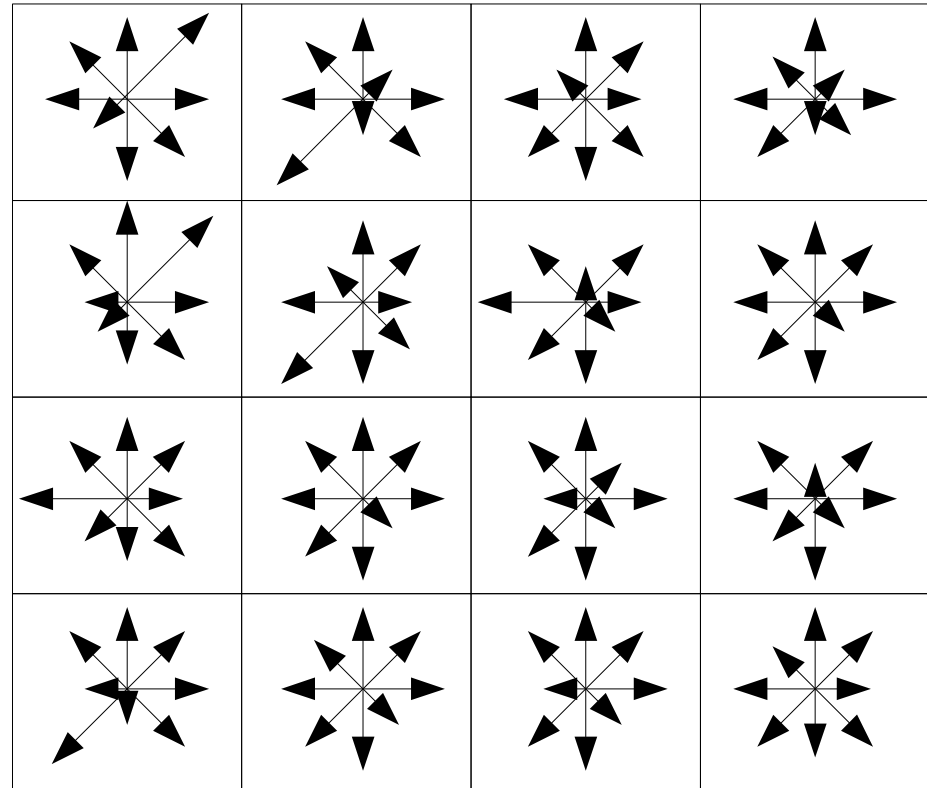
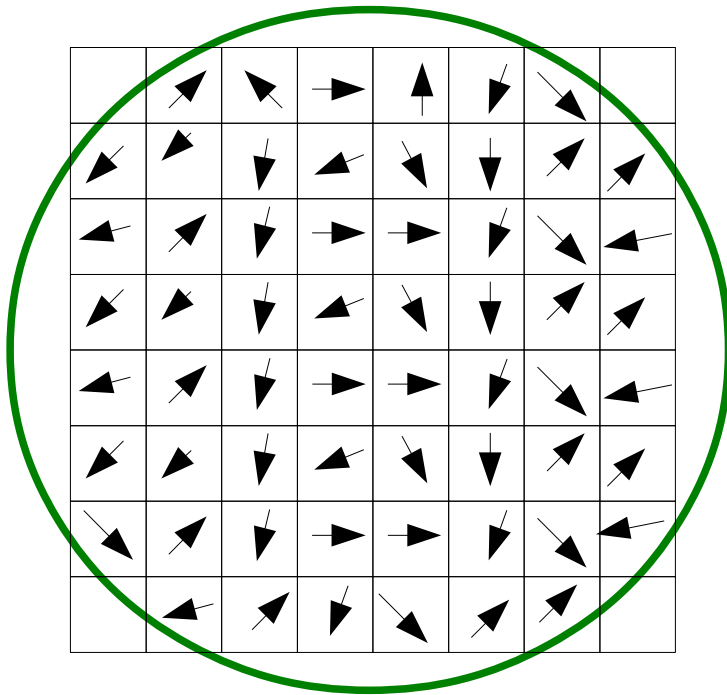
Set of orientation histograms are computed on 4x4 pixel areas.

Each gradient histogram contains 8 bins and each descriptor contains an array of 4 histograms.

→ SIFT descriptor as 128 ($4 \times 4 \times 8$) element histogram



Visualising the keypoint descriptor





Example SIFT keypoints





- Alternative to SIFT - “Speeded Up Robust Features”
- High dimensionality of SIFT descriptor makes it costly to compute and slow to match.
- Goal is to speed up the detection and description process for image features.
- Similar to SIFT but the authors claim better and more robust performance.



- Uses integral images (similar to summed area tables) to quickly compute box-type convolution filters.
- Integral image = the sum of the intensities of all pixels contained in the rectangle defined by the pixel of interest and the origin.



The value of the integral image at point (x,y) = the sum of all pixels above
and to the left.



$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + i(x, y)$$

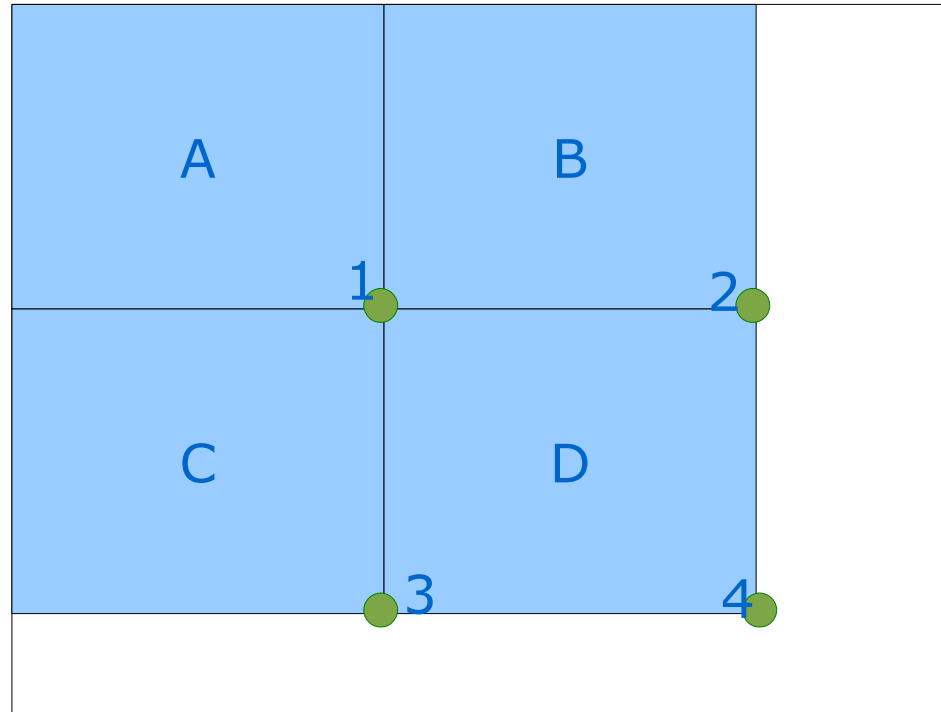
$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

Where **$s(\mathbf{x}, \mathbf{y})$** is the cumulative row sum

$s(\mathbf{x}, -\mathbf{1}) = \mathbf{0}$ and

$ii(-\mathbf{1}, \mathbf{y}) = \mathbf{0}$

the integral image can be computed in one pass
 over the original image



Integral image at point 1 = **sum of pixels in A.**

Value at point 2 = **A+B.**

Value at point 3 = **A+C.**

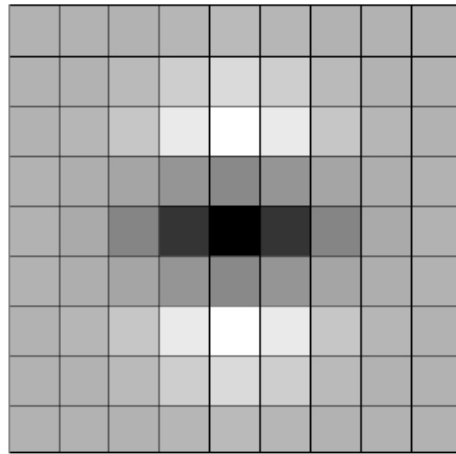
Value at point 4 = **A+B+C+D.**

Sum within D can be calculated as $4 + 1 - (2 + 3)$.



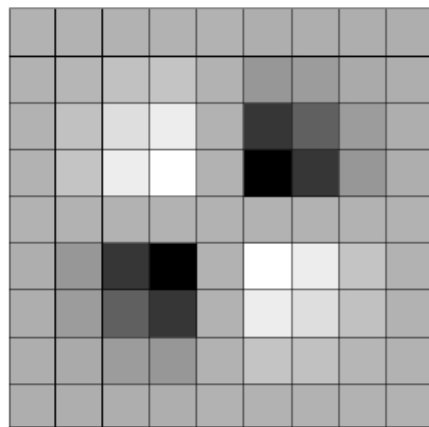
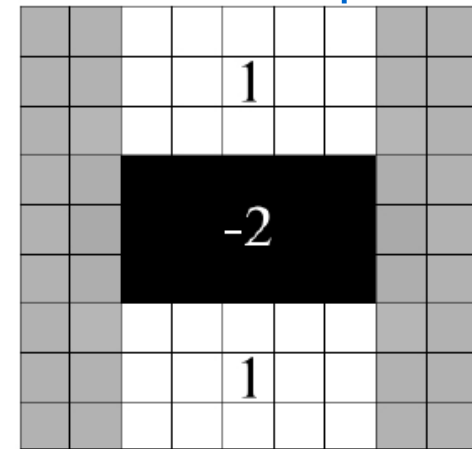
SURF detector

Gaussians

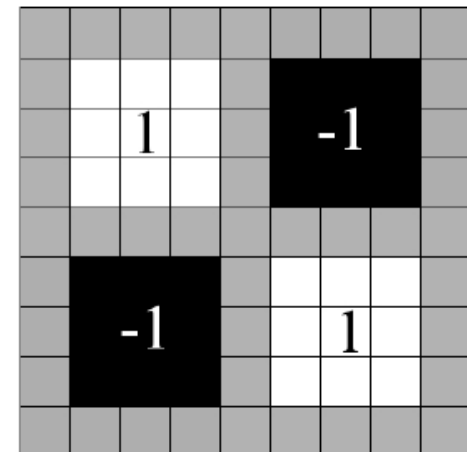


Y direction

Box filter equivalent



XY direction

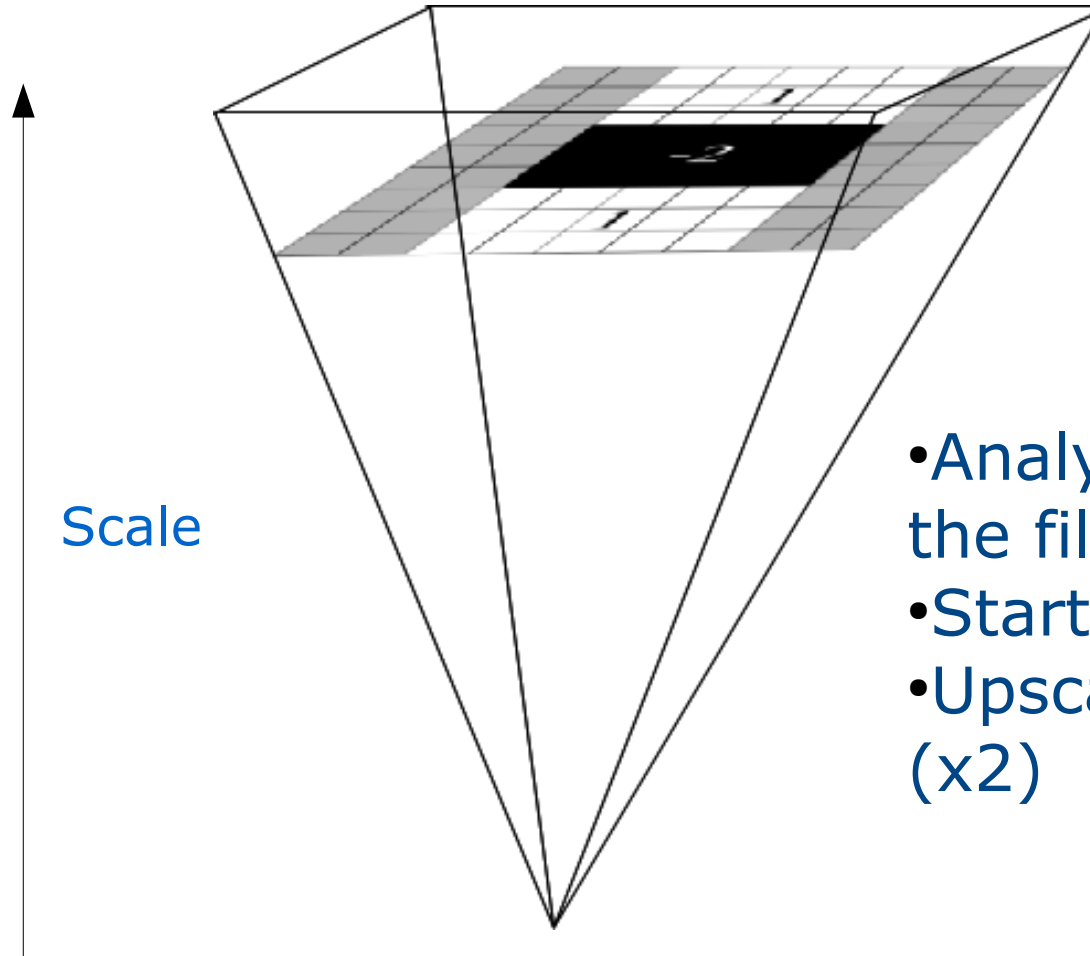


Computation time increases with filter size.

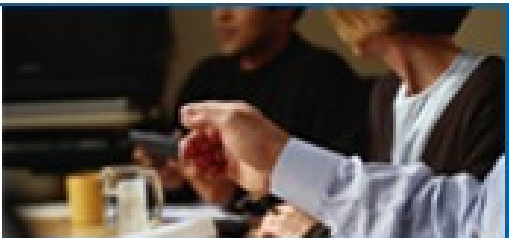
Computation time constant and Independent of filter size.



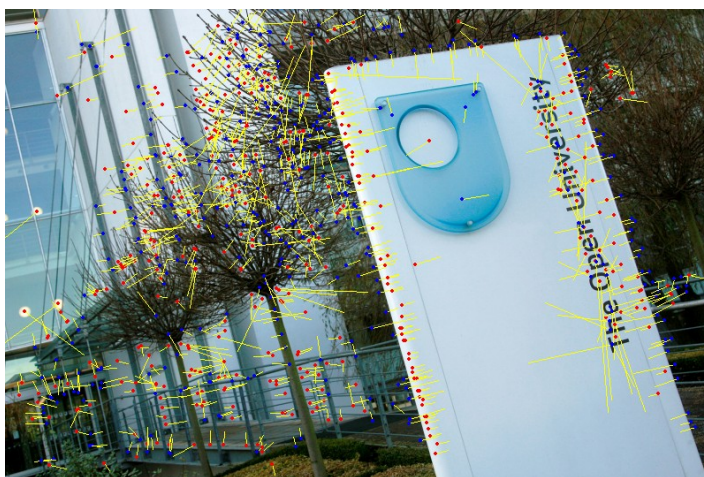
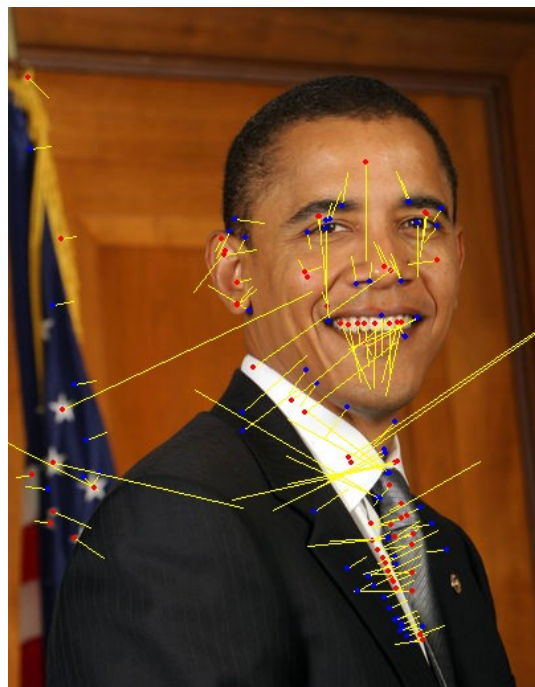
SURF Scale space



- Analyse by upscaling the filter size
- Start with 9x9
- Upscale by octaves (x2)



SURF: some example images





SURF and SIFT both focus on the spatial distribution of gradient information.

SURF

Is three times faster than SIFT

Is less susceptible to noise (claimed to be!)

Is good at handling serious image blur

Is good at handling image rotation

Does not handle viewpoint change or illumination change well

NB: SURF does not always outperform the original SIFT