Spherical Hashing

Jae-Pil Heo[1], Youngwoon Lee[1], Junfeng He[2], Shih-Fu Chang[2], and Sung-Eui Yoon[1]

[1]KAIST [2]Columbia Univ.

IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2012

Introduction

- Approximate *k*-nearest neighbor search in high dimensional space
  - widely used in various applications
  - high computation cost, memory requirement
  - tree-based methods do not give any benefit (*curse of dimensionality*)
  - spatial hashing techniques get more attention

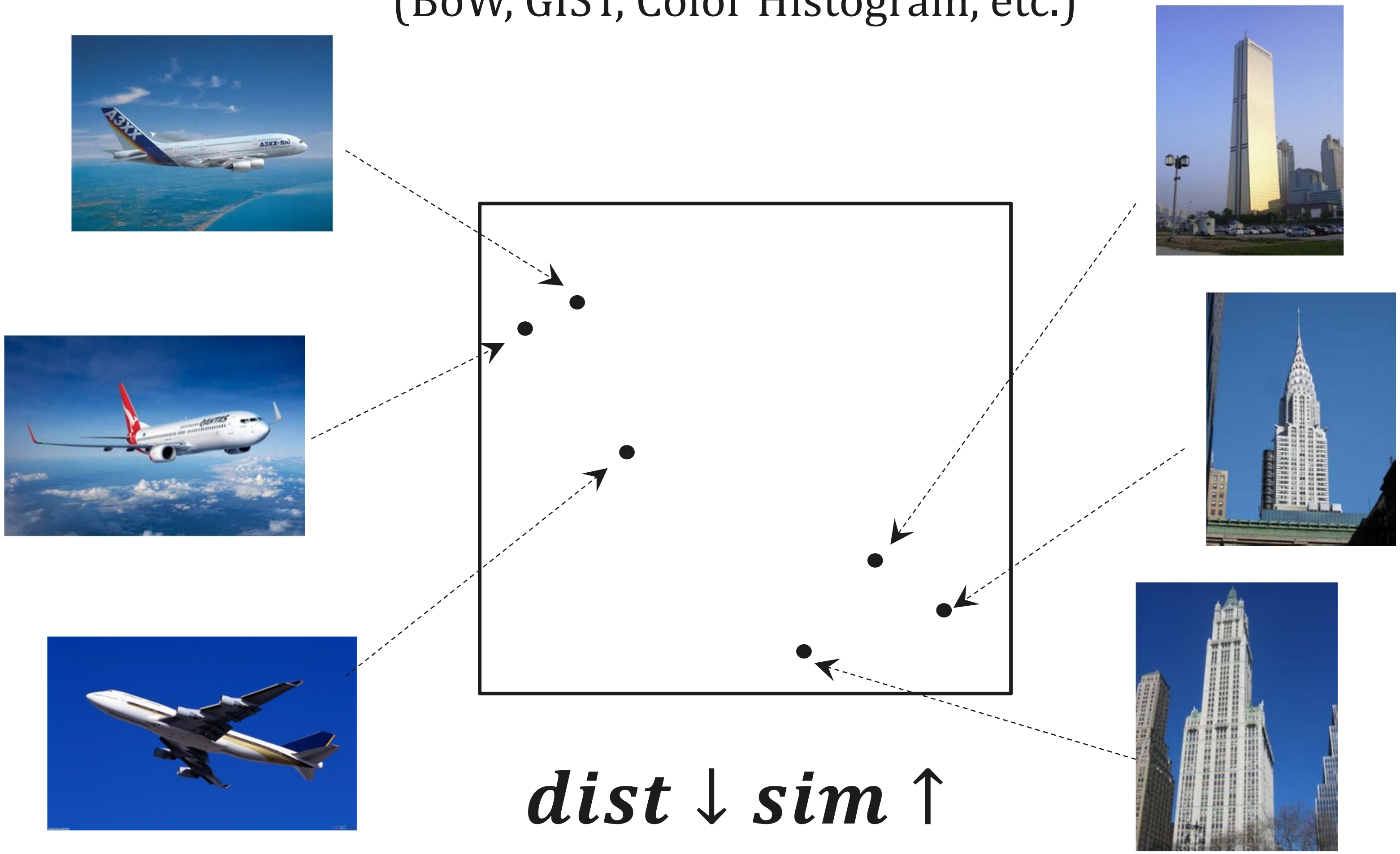
Image Retrieval

Finding visually similar images

Image Descriptors

High dimensional point

(BoW, GIST, Color Histogram, etc.)

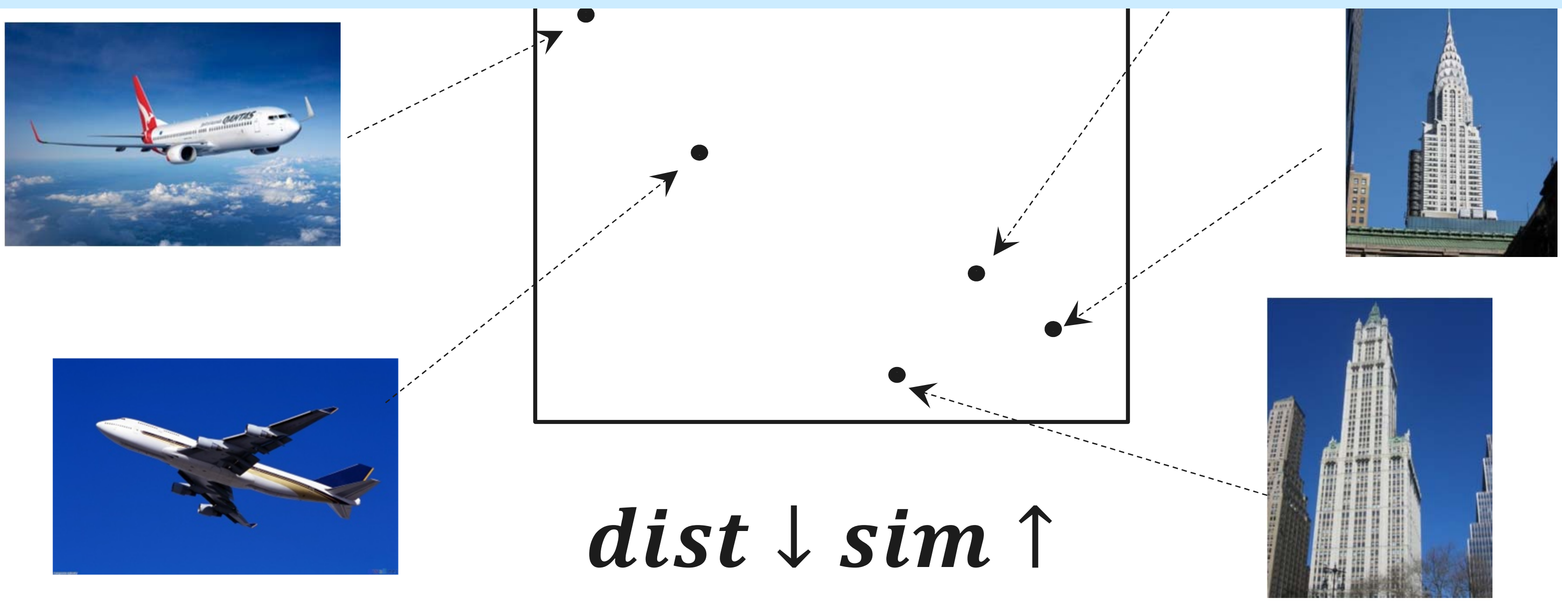


$dist \downarrow sim \uparrow$

Image Descriptors

High dimensional point

Image retrieval is reduced to nearest neighbor search in high dimensional space



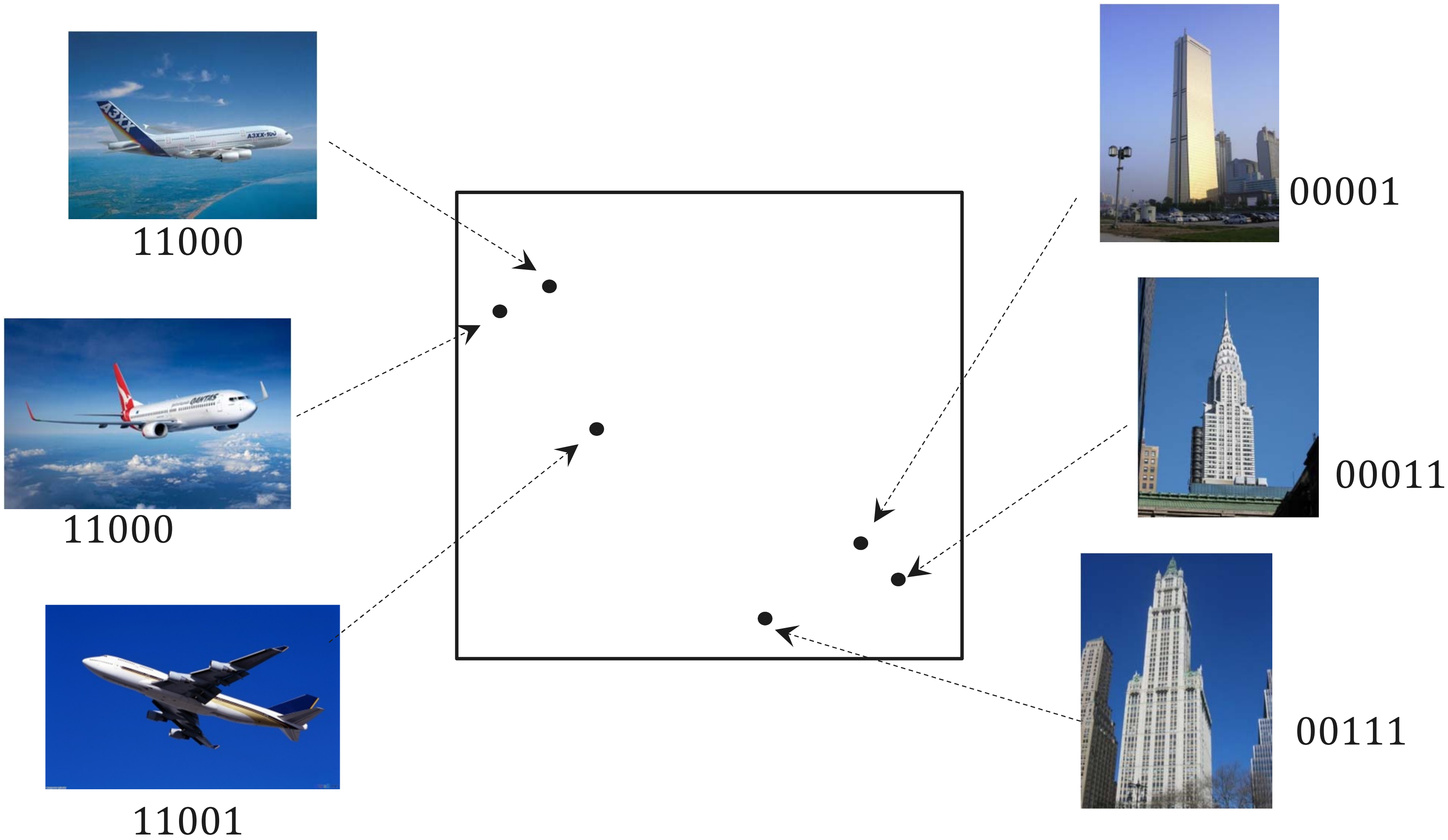
$dist \downarrow sim \uparrow$

Challenge

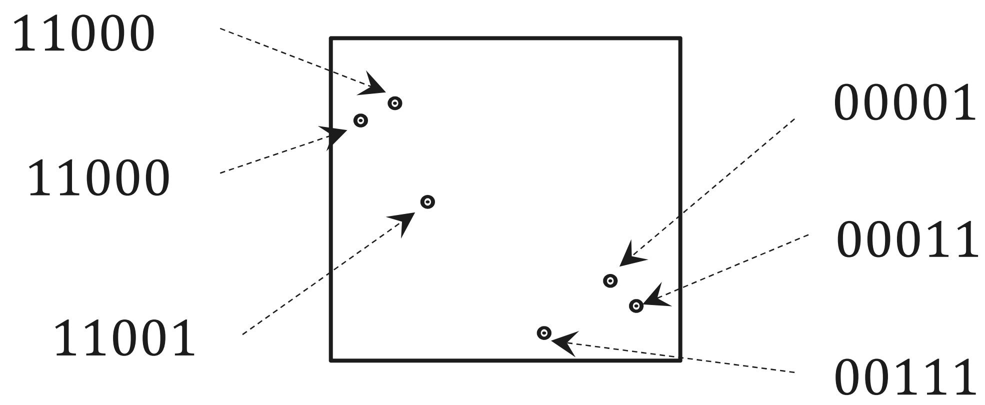
	BoW	GIST
Dim	1000+	300+
1 image	4 KB+	1.2 KB+
1B images	3 TB+	1 TB+

$$\frac{144\ GB\ memory}{1\ billion\ images} \approx \frac{128\ bits}{1\ image}$$

Binary Codes



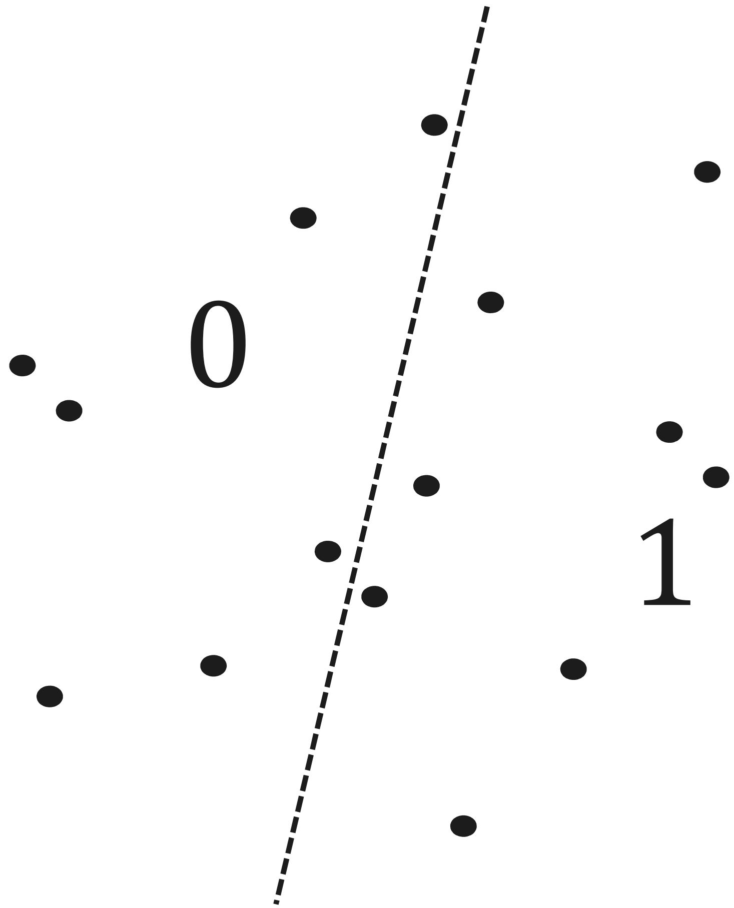
Binary Codes



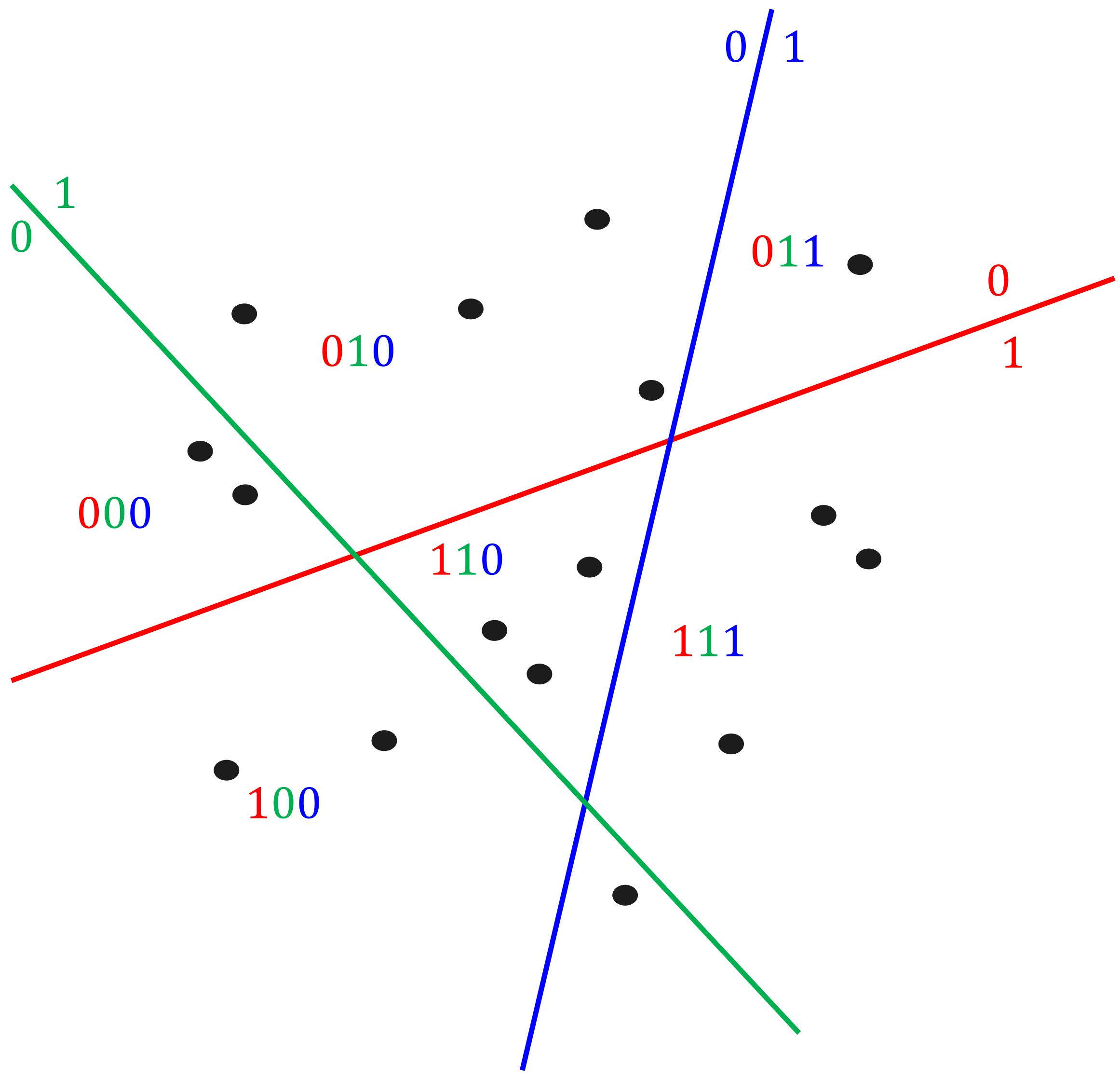
* Benefits
- High compression ratio (scalability)
- Fast similarity calculation with Hamming distance (efficiency)

*** Issue**
- **How well do binary codes preserve data positions and their distances (accuracy)**

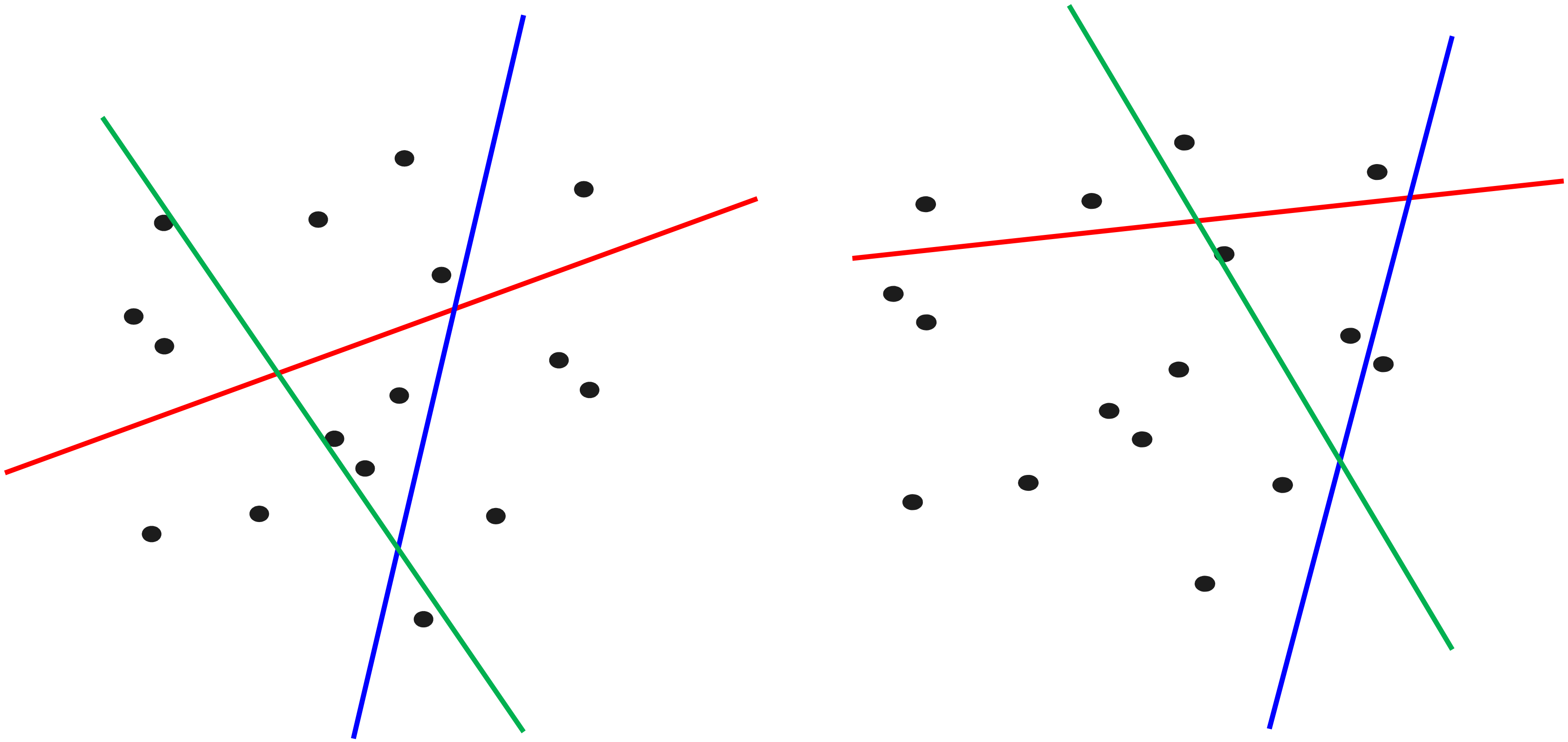
Binary Code with Hyper-Planes



Binary Code with Hyper-Planes



Good and Bad Hyper-Planes



Previous work focused on
how to determine good hyper-planes

State-of-the-art Methods

- Random hyper-planes from a specific distribution
  [Indyk – STOC 1998, Raginsky – NIPS 2009]
- Spectral graph partitioning
  [Yeiss – NIPS 2008]
- Minimizing quantization error (ITQ)
  [Gong – CVPR 2011]
- Independent component analysis (ICA)
  [He – CVPR 2011]
- Support vector machine (SVM)
  [Joly – CVPR 2011]

- **All of them use hyper-planes!**

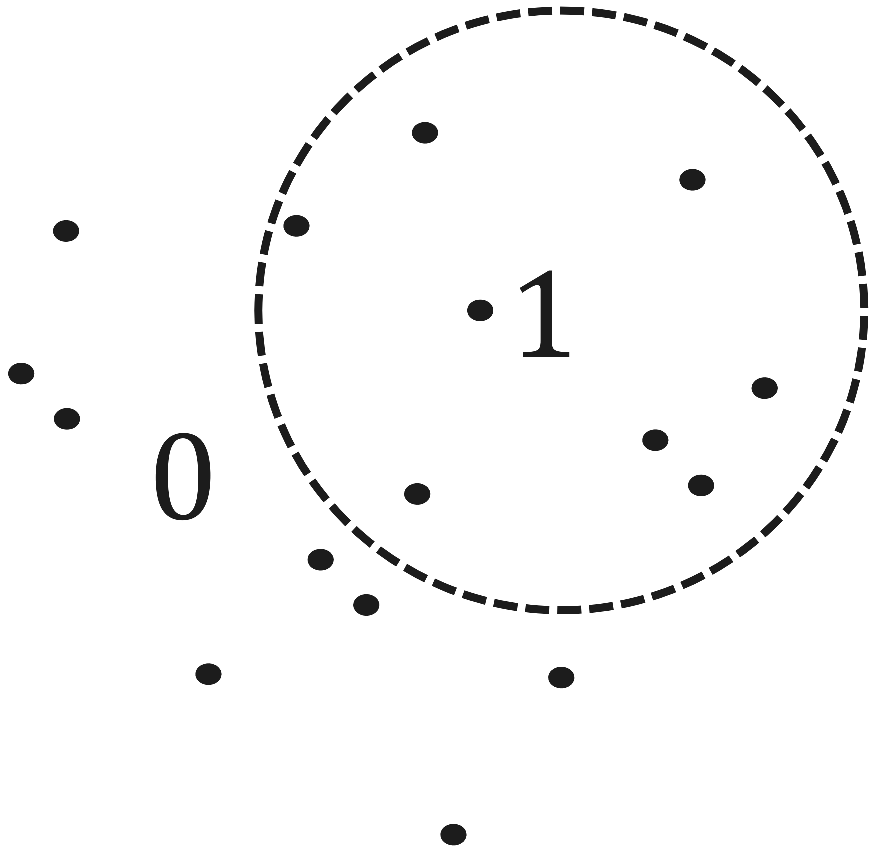
Our Contributions

- Spherical Hashing
- Iterative optimization scheme to determine hyper-spheres
- Spherical Hamming distance

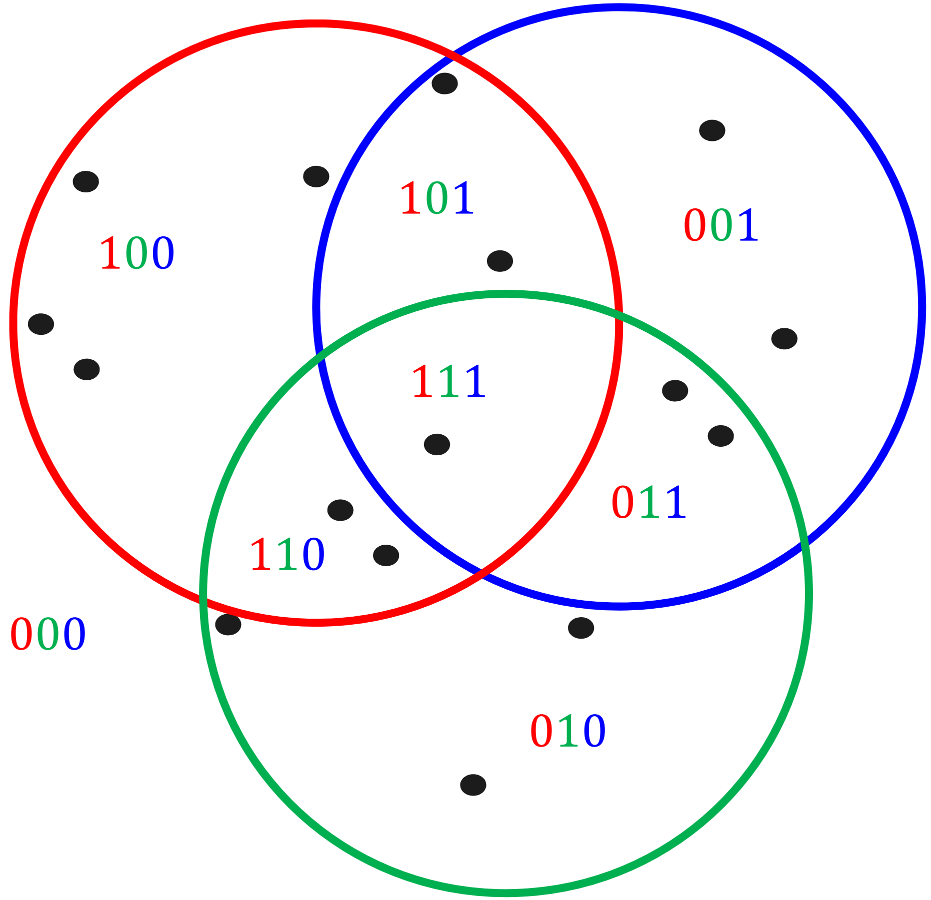
Our Contributions

- Spherical Hashing
- Iterative optimization scheme to determine hyper-spheres
- Spherical Hamming distance

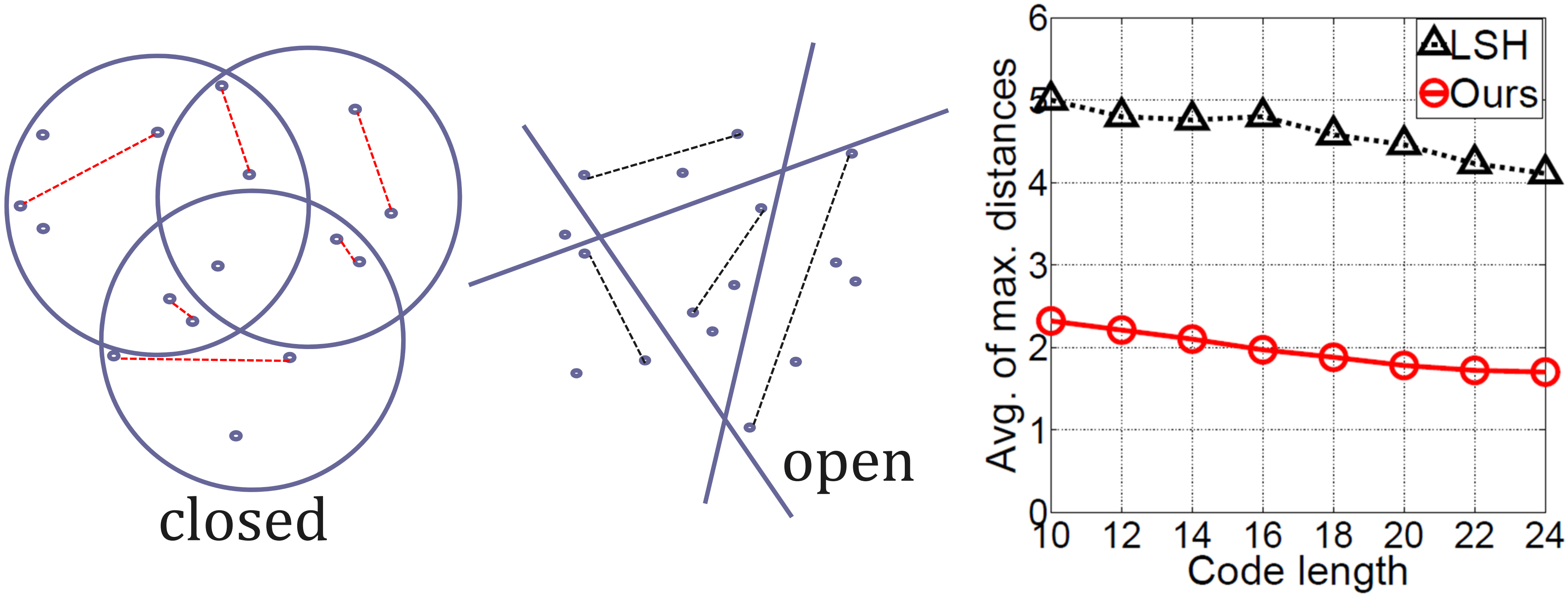
Spherical Hashing



Partitioning Example



Bounding Power of Hyper-Sphere



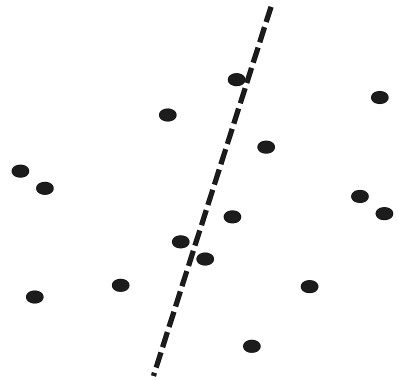
Average of maximum distances within a partition:
- Hyper-spheres gives tighter bound!

Our Contributions

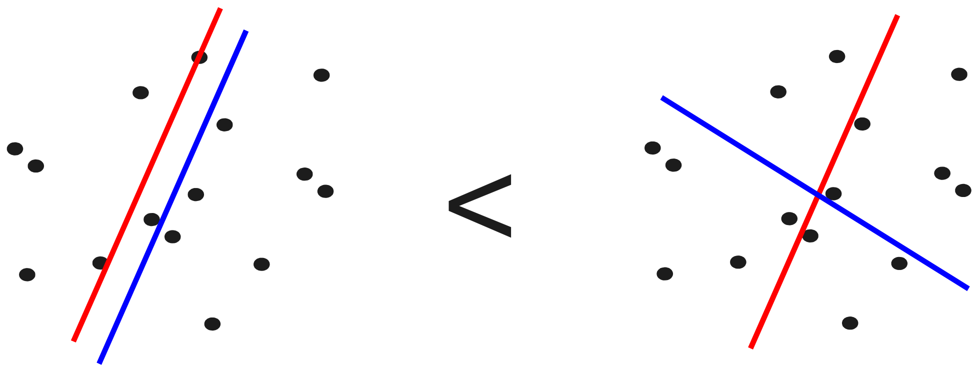
- Spherical Hashing
- Iterative optimization scheme to determine hyper-spheres
- Spherical Hamming distance

Two Criteria [Yeiss 2008, He 2011]

1. Balanced partitioning

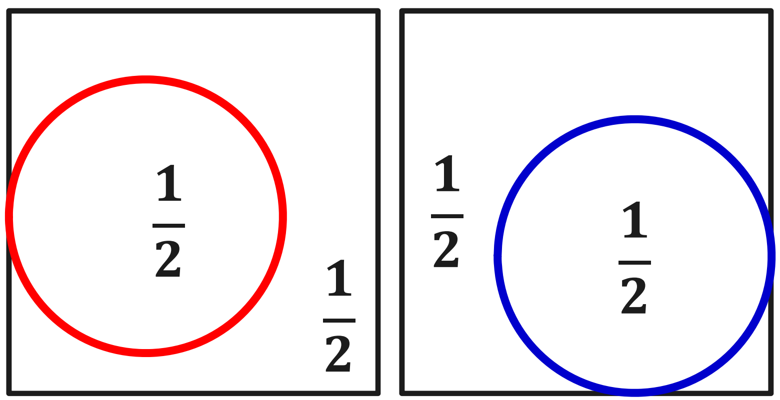


2. Independence

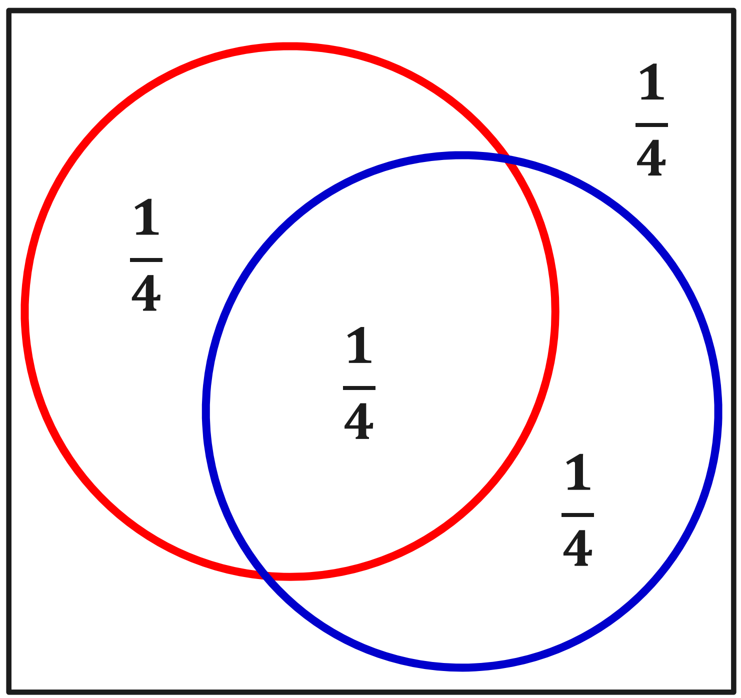


Two Criteria with Hyper-Spheres

1. Balance

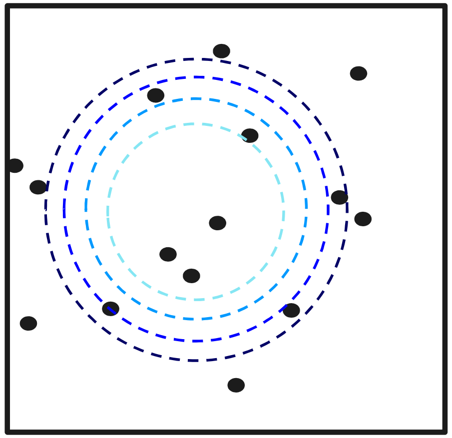


2. Independence

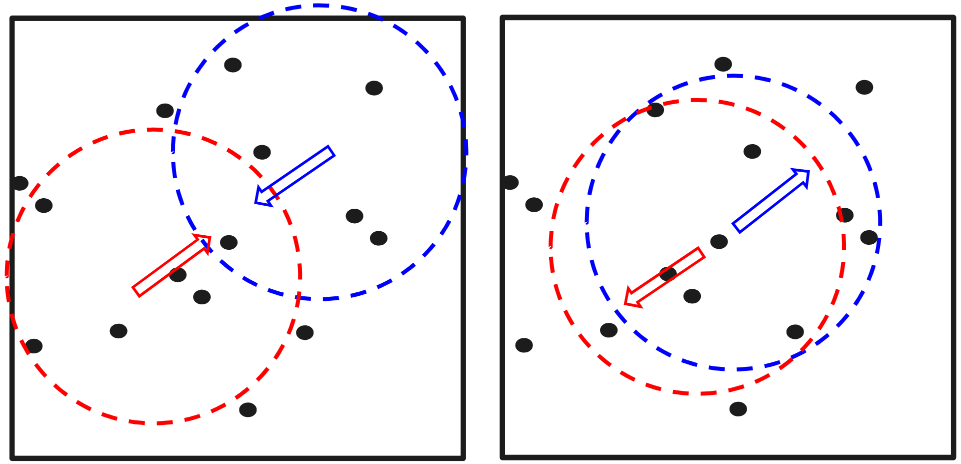


Iterative Optimization

1. Balance
- by controlling radius
for $\boldsymbol{n}(\boldsymbol{S}) = \frac{\boldsymbol{N}}{\boldsymbol{2}}$



2. Independence
- by moving two hyper-spheres
for $\boldsymbol{n}(\boldsymbol{S_1} \cap \boldsymbol{S_2}) = \frac{\boldsymbol{N}}{\boldsymbol{4}}$

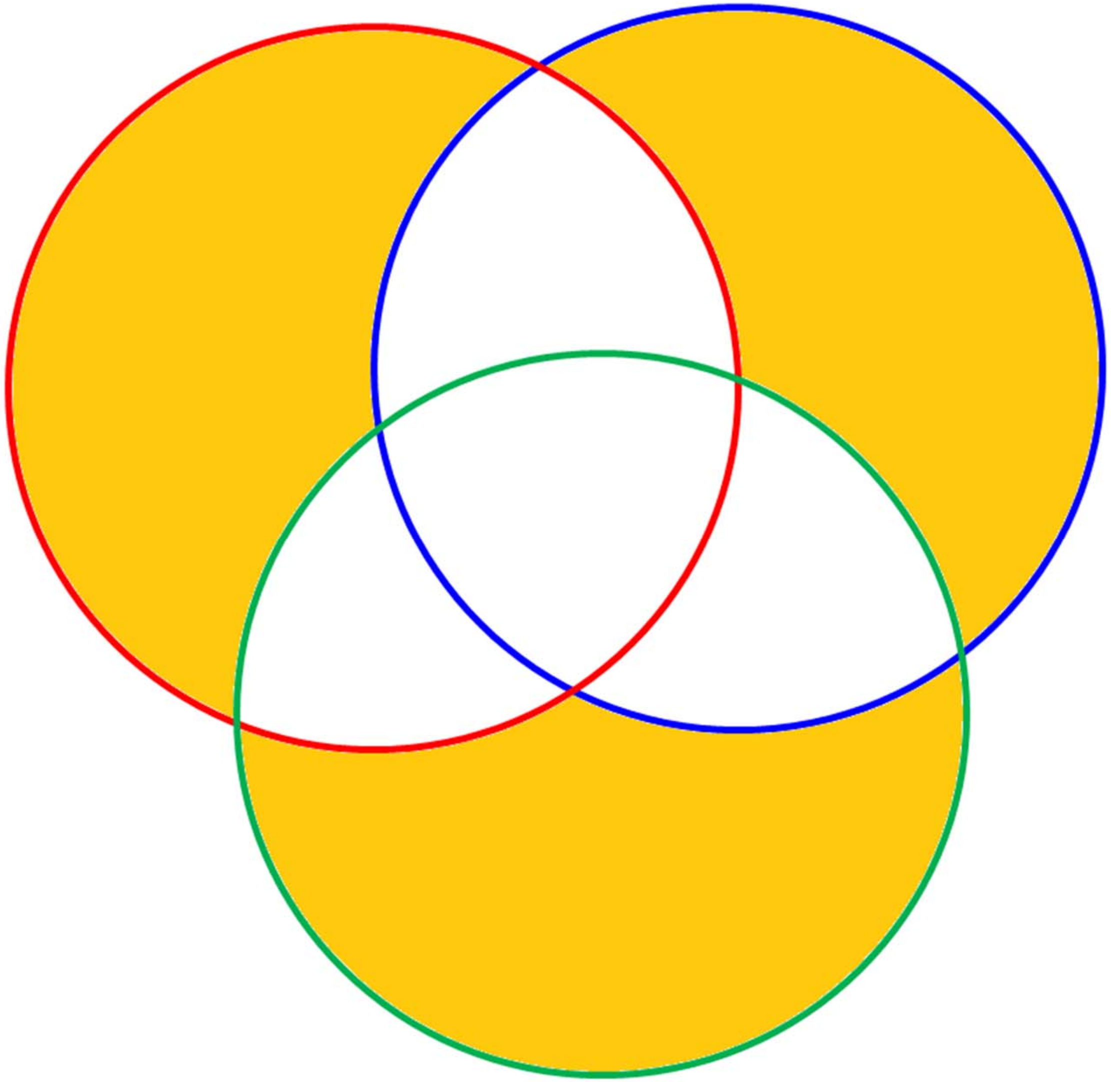


Repeat step 1, 2 until convergence.

Our Contributions

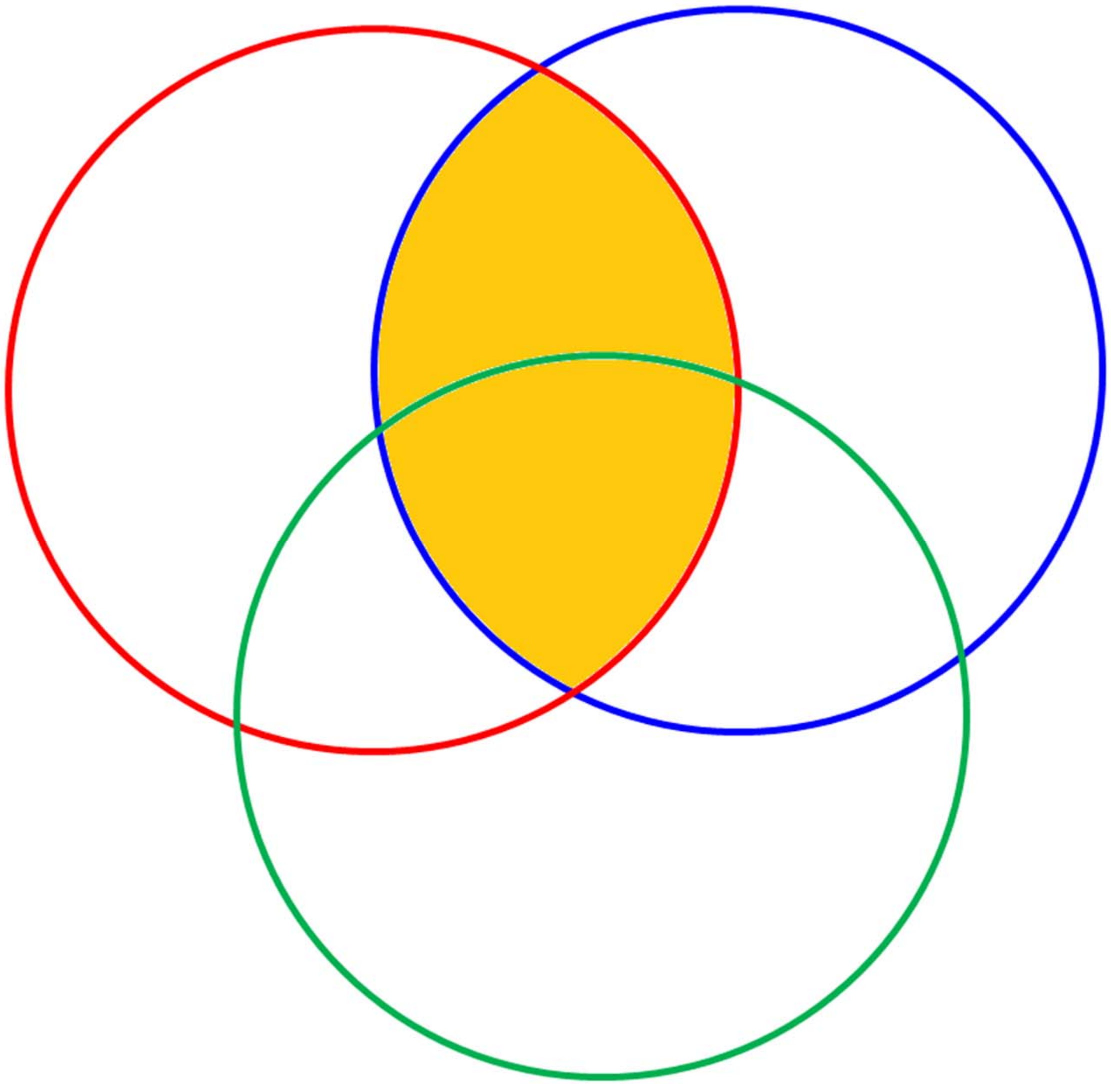
- Spherical Hashing
- Iterative optimization scheme to determine hyper-spheres
- Spherical Hamming distance

Intuition of Spherical HD



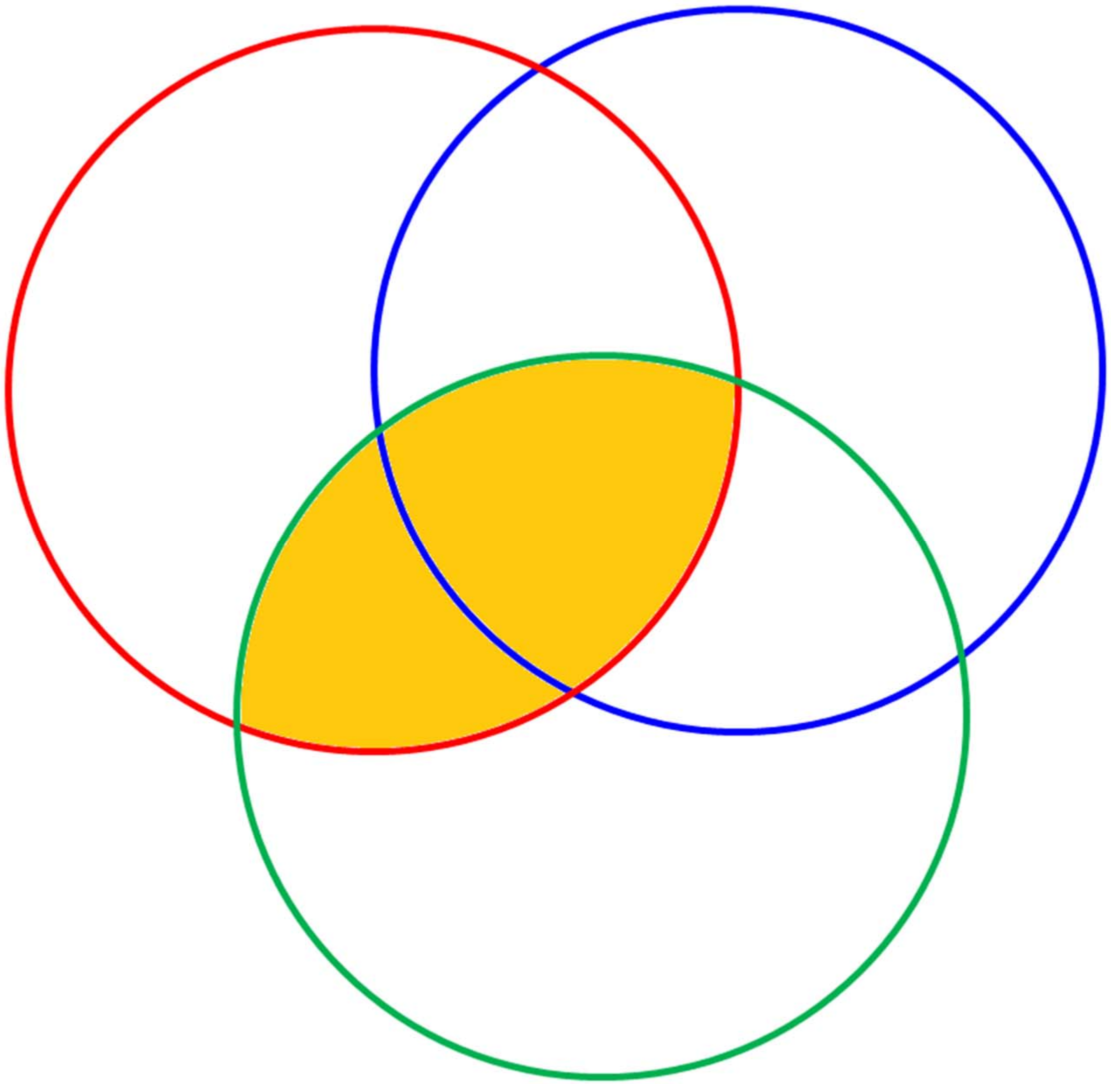
Bounded by 1 hyper-sphere

Intuition of Spherical HD



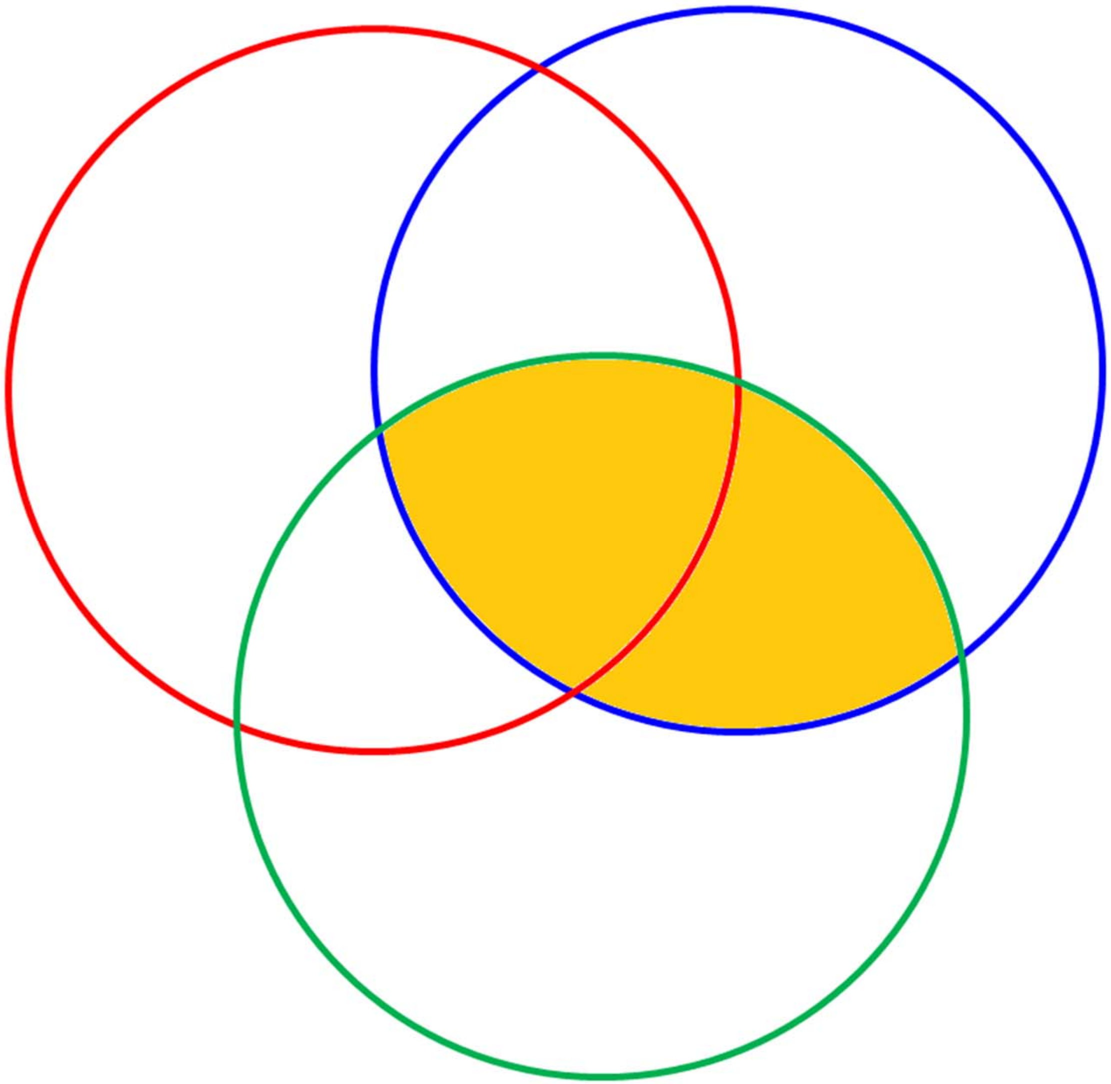
Bounded by 2 hyper-spheres

Intuition of Spherical HD



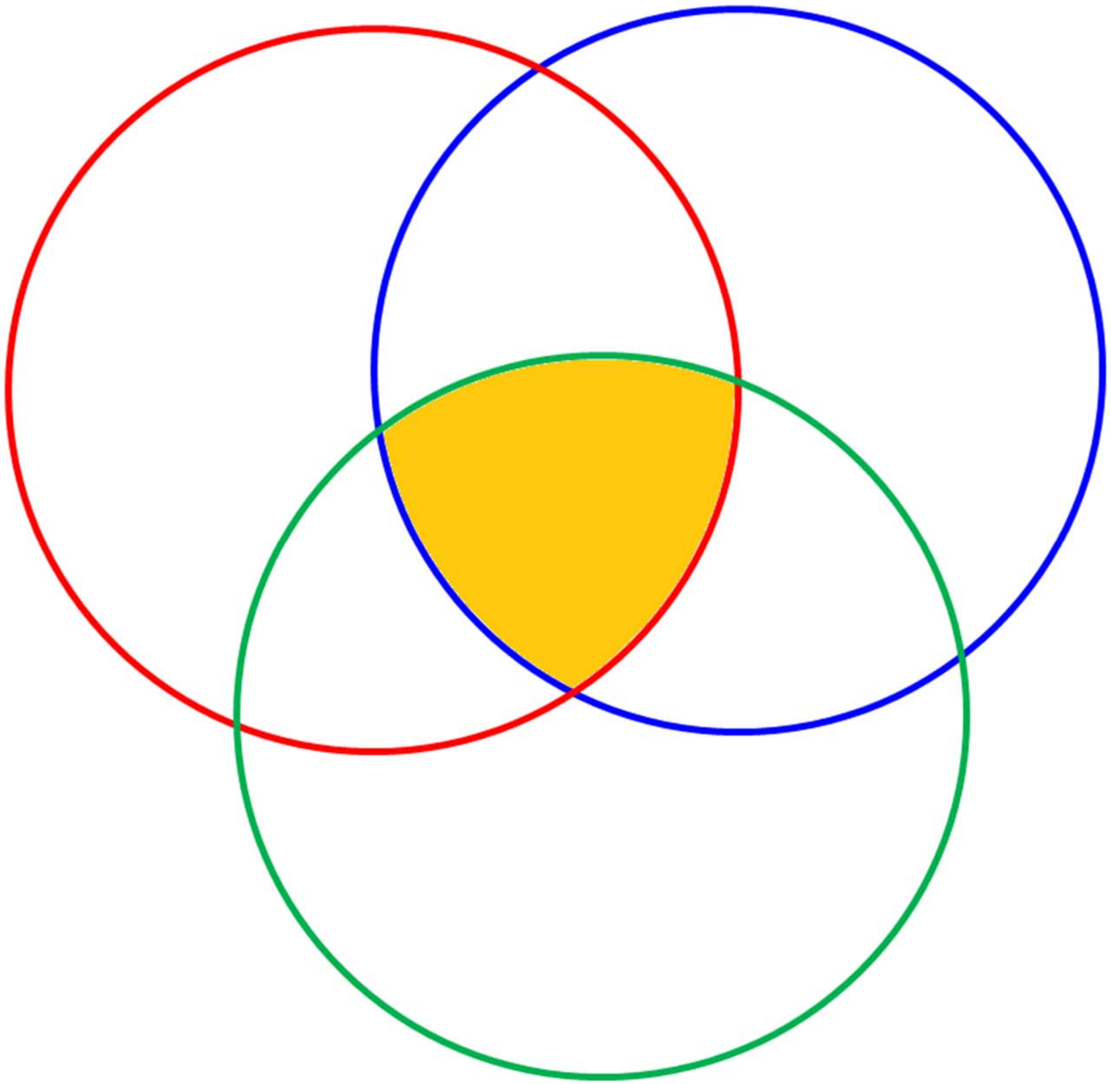
Bounded by 2 hyper-spheres

Intuition of Spherical HD



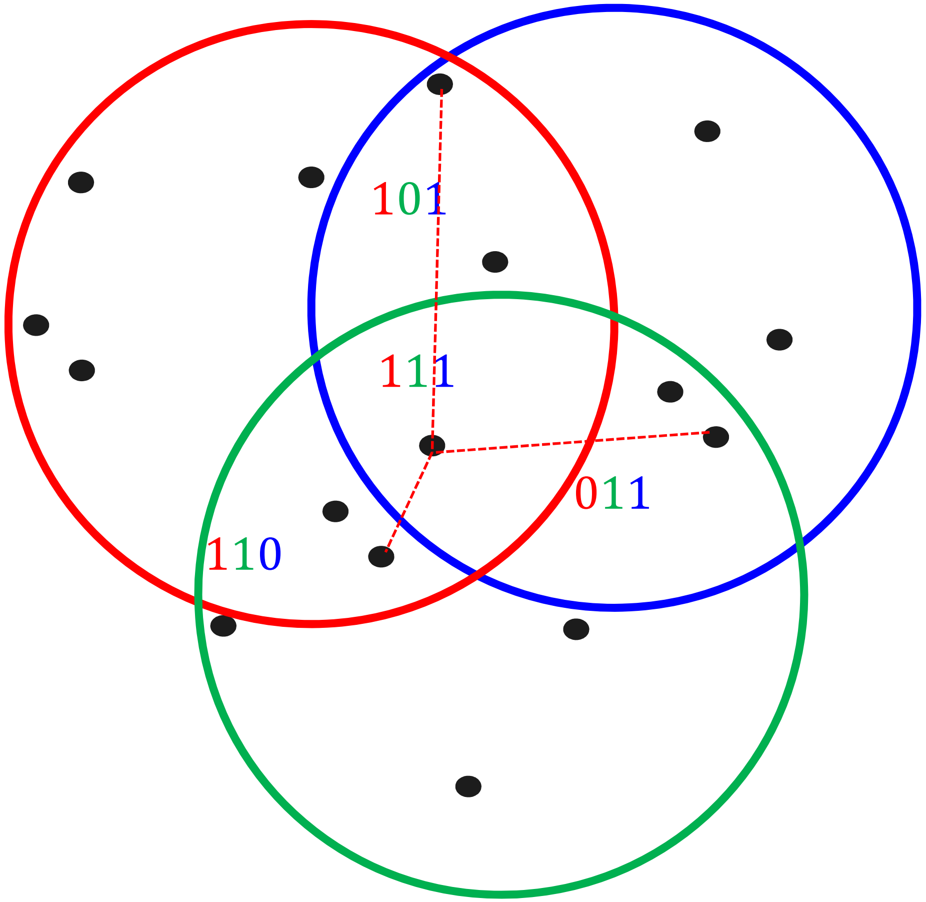
Bounded by 2 hyper-spheres

Intuition of Spherical HD



Bounded by 3 hyper-spheres

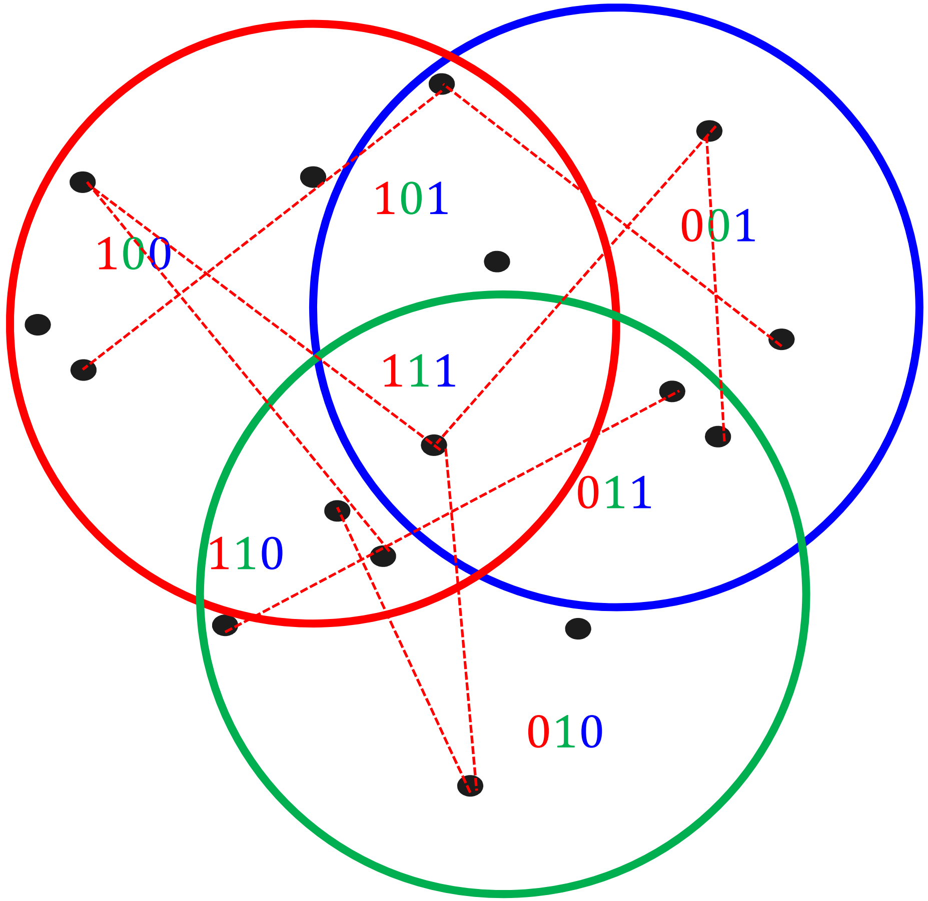
Max Dist. and Common ‘1’



Common ‘1’s

: 2

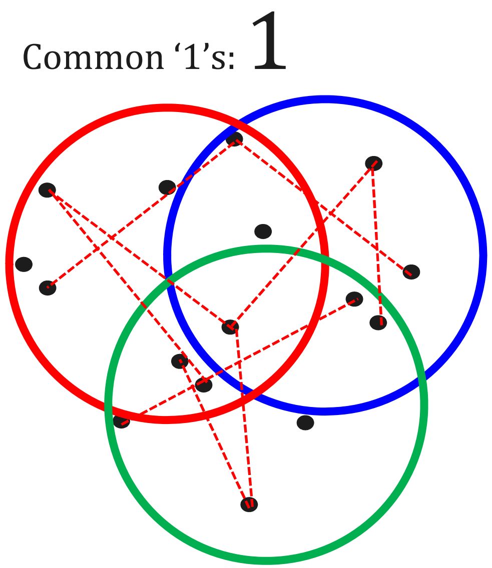
Max Dist. and Common '1'

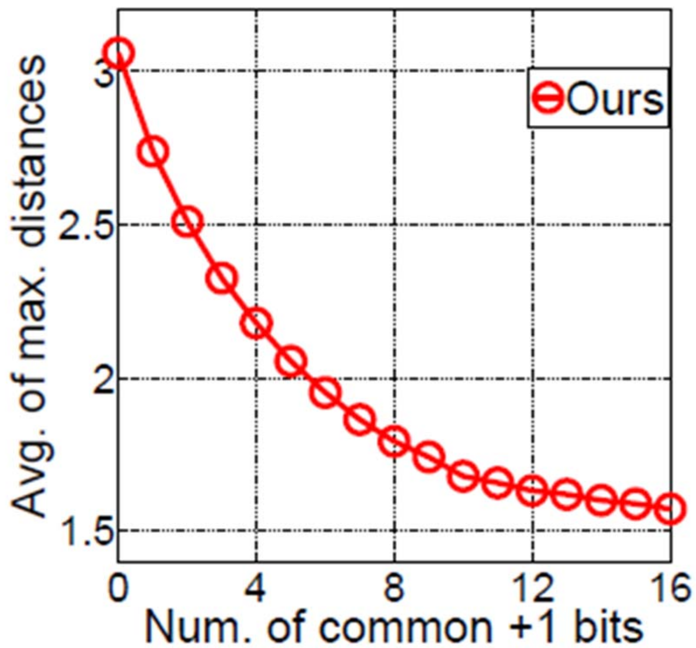


Common '1's

: 1

Max Dist. and Common '1'





Average of maximum distances between two partitions: decreases as number of common '1'

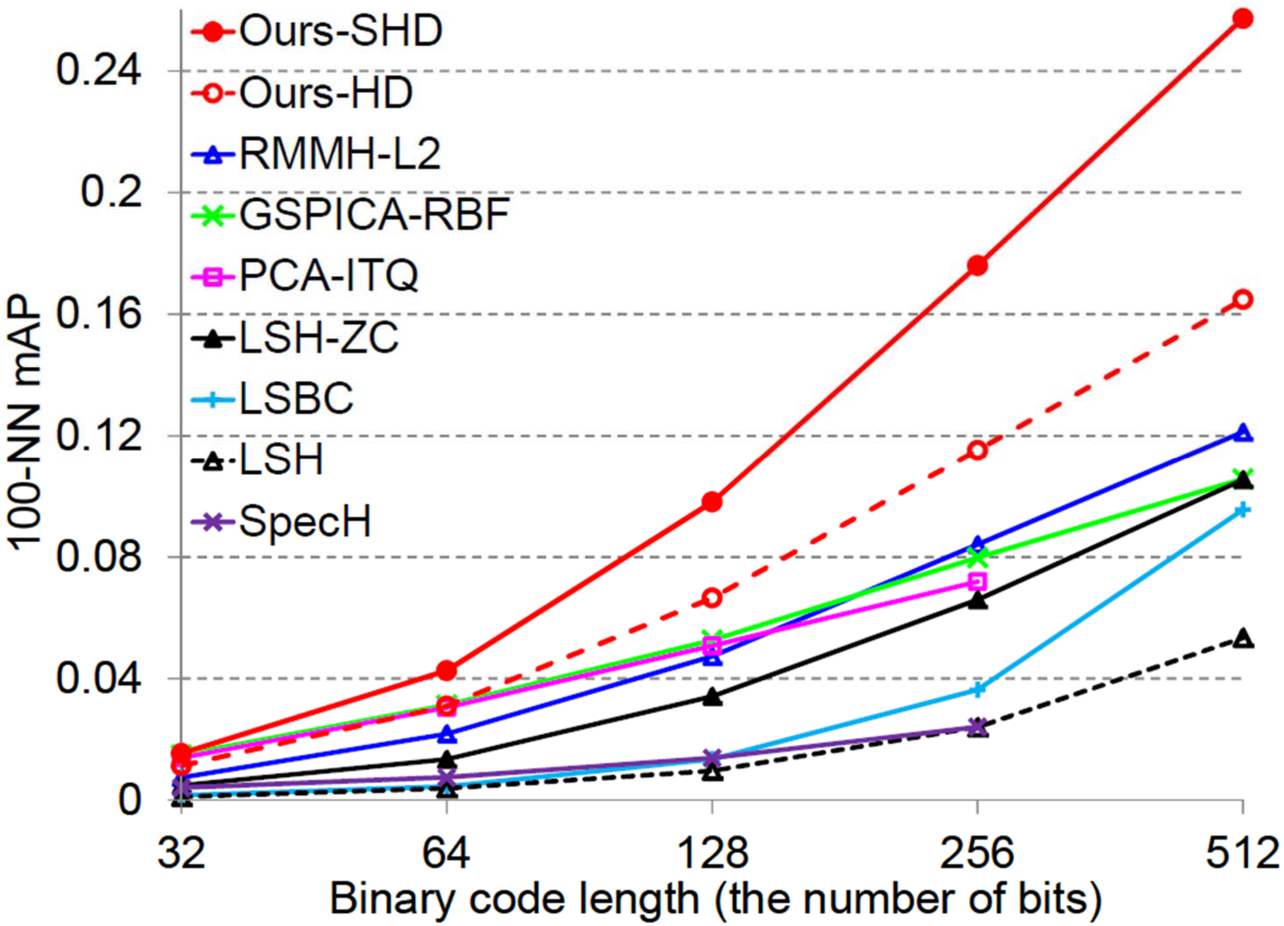
Spherical Hamming Distance (SHD)

$$d_{shd}(b_i, b_j) = \frac{|b_i \oplus b_j|}{|b_i \wedge b_j|}$$

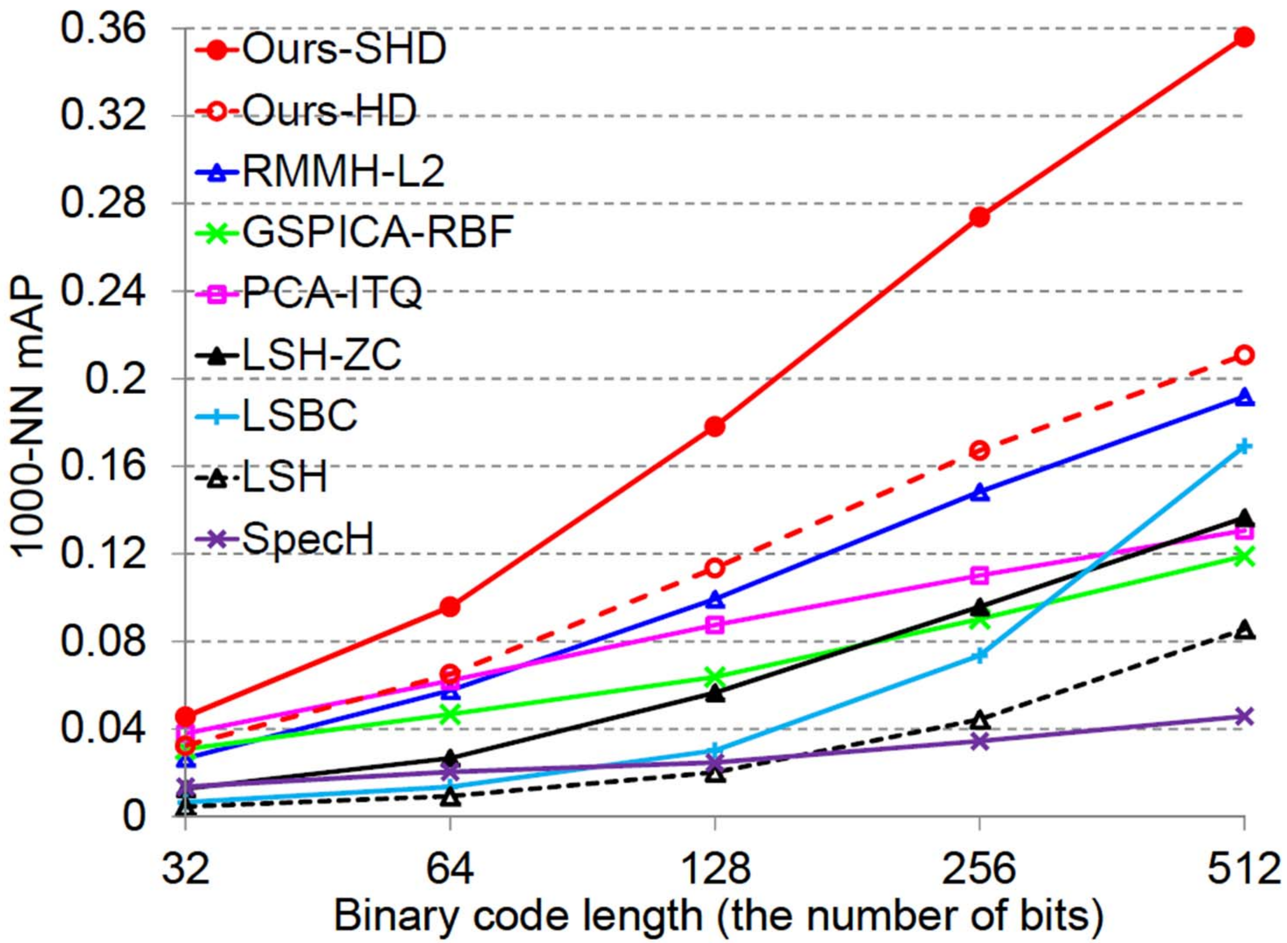
SHD: Hamming Distance divided by the number of common ‘1’s.

b_i: *binary code* $\quad \oplus$: *XOR* $\quad \wedge$: *AND*

Result (1M, 384 dim GIST)



Result (1M, 960 dim GIST)



Result (75M, 384 dim GIST)