Robust Image Hashing Using Image Normalization and SVD Decomposition

Ricardo Antonio Parrao Hernandez[1]
Brian M. Kurkoski[2]
Mariko Nakano Miyatake[1]

[1]National Polytechnic Institute, Mexico
[2]University of Electro Communications, Tokyo, Japan

Importance of Image Authentication

- Digital images are all around us
  - Very easy to manipulate images
- Digital images are used as an evidence:
  - legal proceedings, medical cases, political scandals
- Digital images widely used in social networks
  - In Mexico 20% of divorce refer to Facebook pictures

Tampering

Two Approaches: Watermarking and Hashes

Goal: certify that an image was not modified in transit

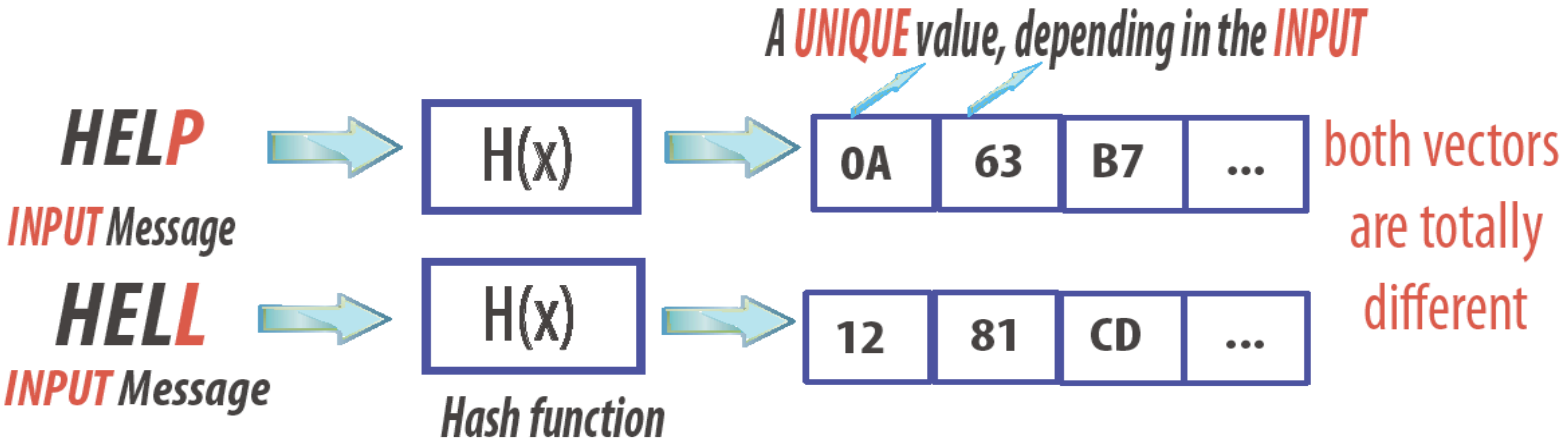
- **Watermarking**:
  - embed authentication information into the image
  - information should be difficult to remove
  - requires modifying the image
- **Image Hashing**
  - Create a numeric signature from the image
  - “Content-based hash functions”
  - does not require modifying an image
  - requires transmitting the signature separately
    - User applies content-based hash function to image
    - If signature is the same, image is authentic

Hash Functions

- A procedure or mathematical function which
  - converts a large, variable-sized amount of data into a small data
- Many applications
  - Finding items in a databases
  - Speed up table lookup
  - Detecting duplicated or similar records in a large file
  - Authentication
- Cryptographic hash functions are widely used
  - verify integrity of files
  - password verification
  - typical algorithms: MD5, SHA1

Cryptographic Hash Functions

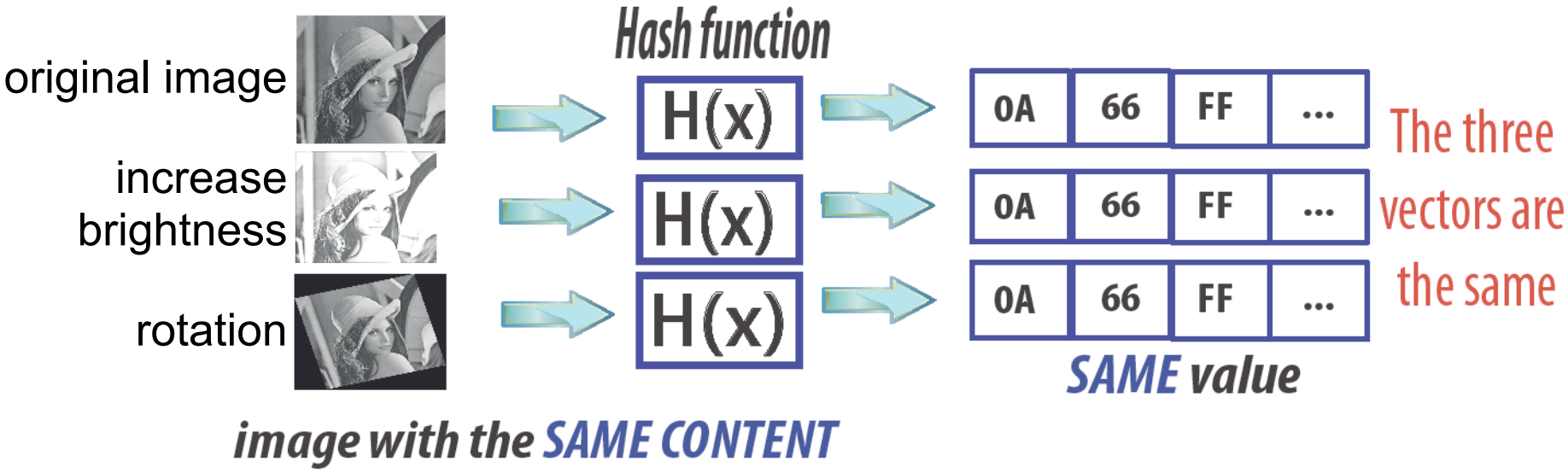
- Input: long, variable-length message
- Output: a short, a fixed-length value



- Cryptographic hashes are bit-sensitive:
  - change of one input bit → output hash value is completely different
  - useful for protecting passwords, etc.
  - not useful for image hashing

Content-Based Hash Functions

- Image Hashing:
  - Visually similar content produces similar hash value



Existing Methods

- Various signal-processing methods generate hash value
  - Using interest point detectors (Harris)
  - Using invariant transforms (FFT, DCT)
  - Using invariant matrices (SVD decomposition)

- Kozat et al gave an SVD-based image hashing algorithm
  - Kozat, Venkatesan and Mihcak “Robust perceptual image hashing via matrix invariants," ICIP 2004
  - SVD decomposition two times,
    - first to subimages of the original image,
    - second to the resulting singular vectors
  - Generally attractive algorithm
  - Tolerates to small changes on rotation until 10°

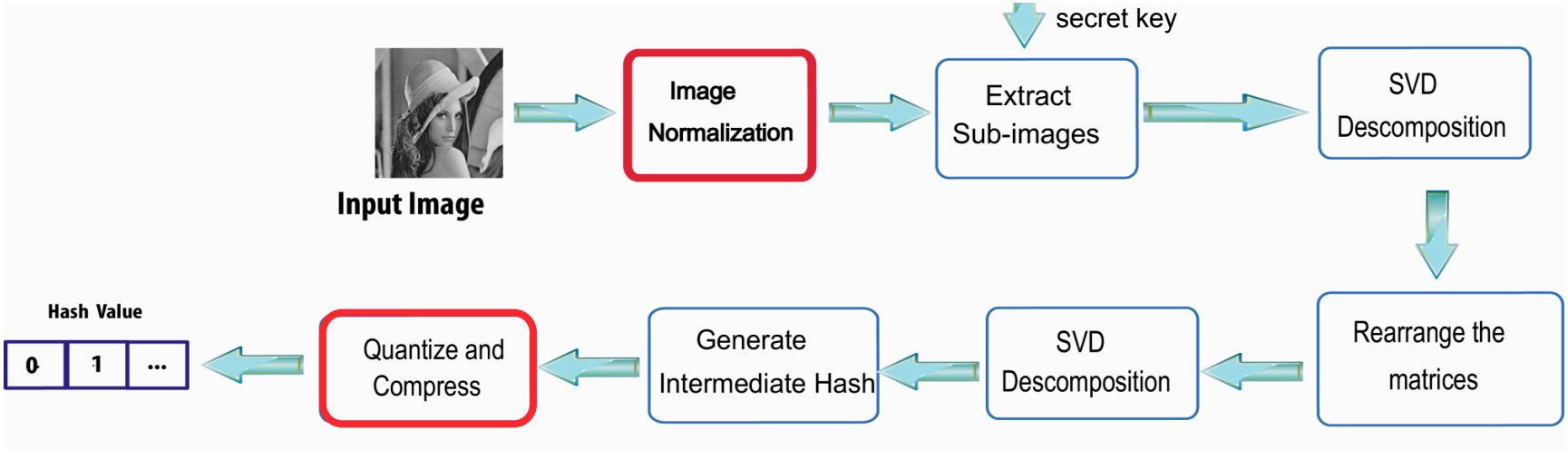
In This Talk....

- Apply **image normalization** to the Kozat et al algorithm to increase the robustness against geometric modifications.
  - Rotated, scaled, etc. images produce the same hash value

Outline

- Proposed algorithm
  - overview (compared to Kozat et al)
  - image normalization
  - Random partition algorithm
  - SVD decomposition
- Numerical evaluation - average Hamming distance
  - rotation
- Conclusions

Proposed Algorithm (versus Kozat et al)



red indicates steps we have added to Kozat's algorithm

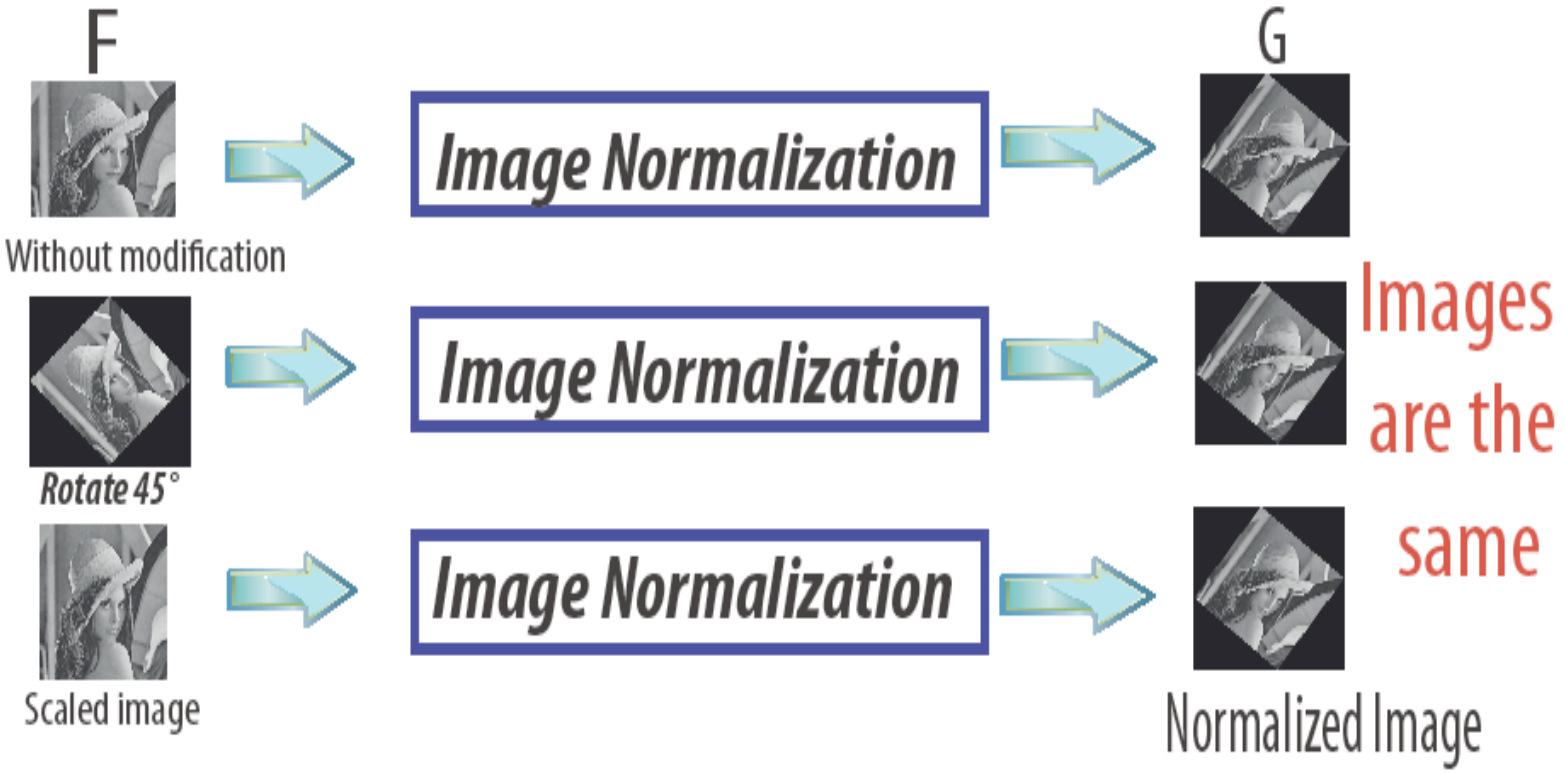
SVD-based hashing

1. image normalization
2. extract sub-images
3. first SVD decomposition
   3. second SVD decomposition
4. intermediate hash
5. quantize and compress

} will explain

1. Image Normalization

- Operations invariant under translation, scaling, orientation
  - Applied to watermarking, Alghoniemy and Tewfix, 2004
- Uses central moments of the image, independent of origin
  - moments are widely used in pattern recognition



1. Image Normalization

The image normalization algorithm has three steps

1) Translation Invariance

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$d_1 = \frac{m_{10}}{m_{00}}, d_2 = \frac{m_{01}}{m_{00}}$$

2) Shearing in the x direction

$$A = \begin{bmatrix} 1 & \beta \\ 0 & 1 \end{bmatrix}$$

$$\mu_{30} + 3\beta\mu_{21} + 3\beta^2\mu_{12} + \beta^3\mu_{03} = 0$$

3) Shearing in the y direction

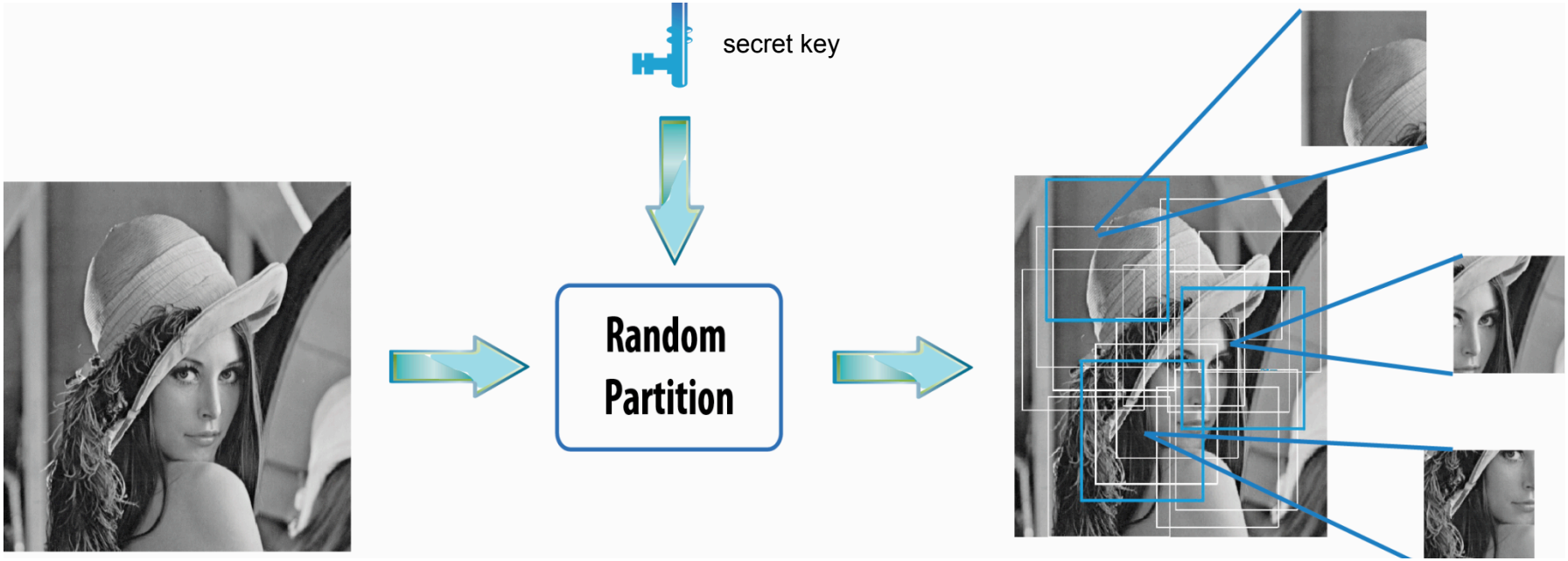
$$A = \begin{bmatrix} 1 & 0 \\ \gamma & 1 \end{bmatrix} \qquad \gamma = -\frac{\mu_{11}}{\mu_{20}}$$

m_{pq} Geometric moments

μ_{pq} Central moments

2. Extraction of Sub-images

- Using random partitioning, extract square sub-images
- A secret key is used to pseudo-randomly select sub-images



3. SVD: Singular-Value Decomposition

- The SVD decomposition of a matrix A is: $A = USV^T$
  - columns of U and V are the singular vectors (content information)
  - diagonal matrix S are singular values (brightness information)
- Image decomposition: $A = \sum_{i=1}^{r} U_i S_i V_i^T$ *r* is the rank of A
- The quality of the reconstruction depends on rank *r*:

Rank 512

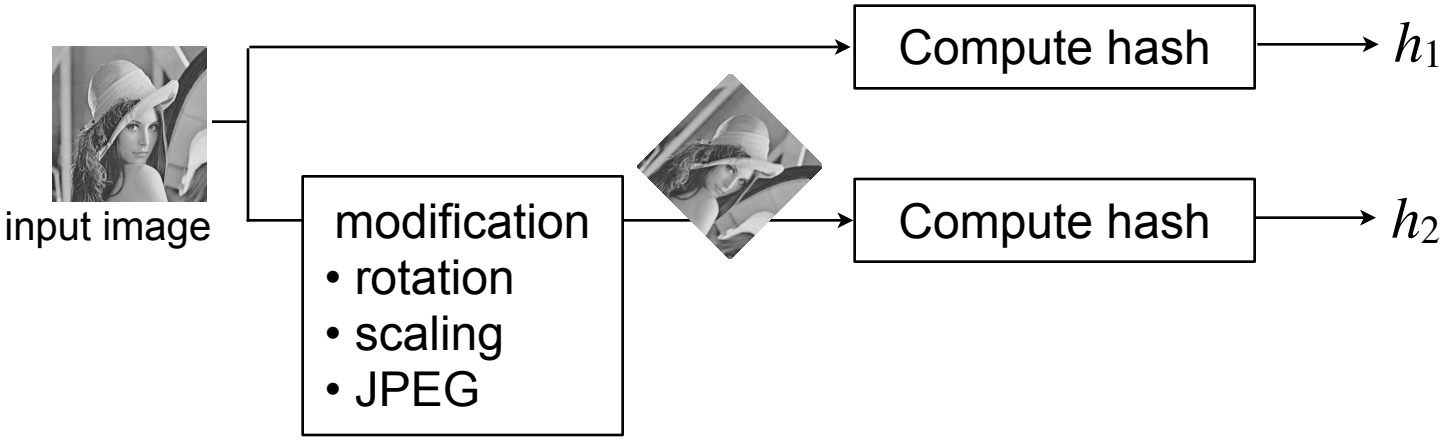
Rank 20

Rank 50

Rank 150

- For image hashing, we take the rank 1 singular vectors
- The second SVD is applied to a matrix of U1 and V1

Numerical Evaluation



- Average Hamming distance between hashes

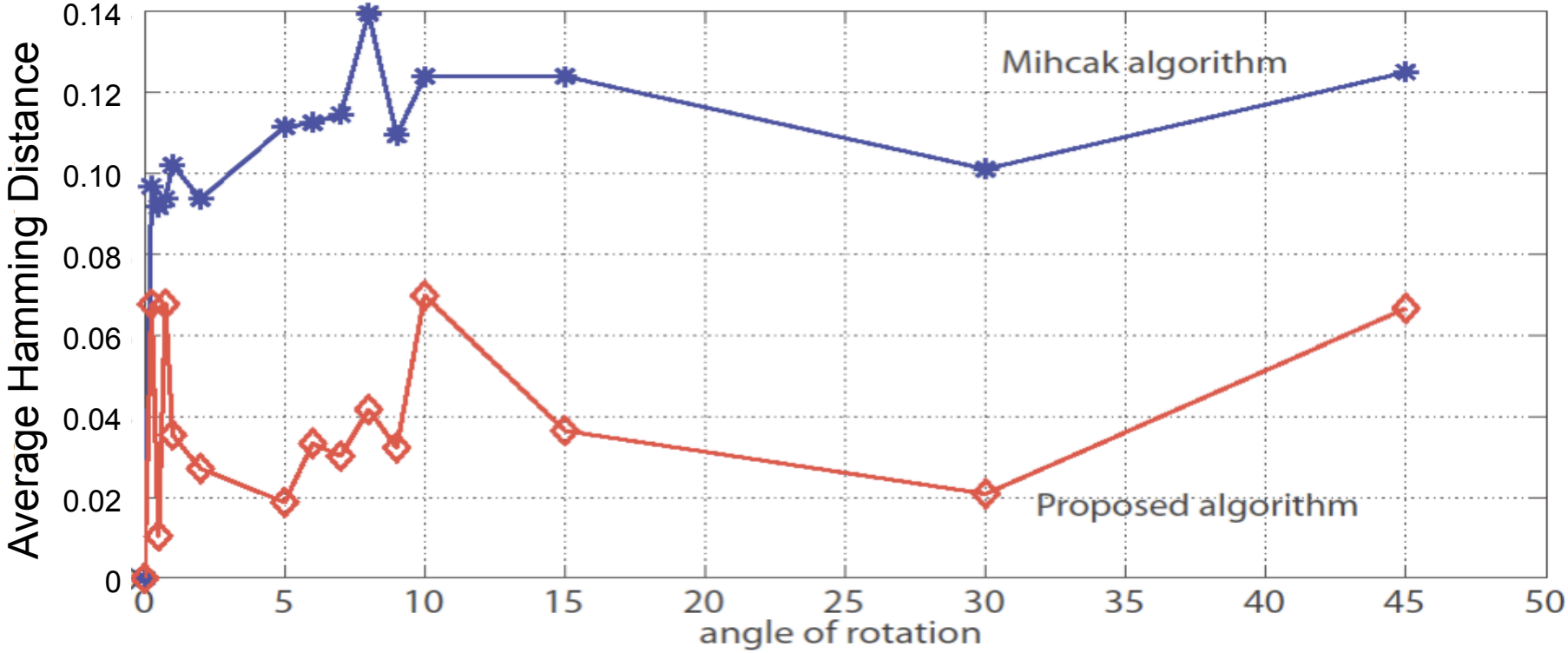
$$d_h(h_1, h_2) = \frac{1}{L}\sum_{k=1}^{L} \|h_1(k) - h_2(k)\|$$

- For the evaluation we use:
  - Graysacle images of size 512-by-512
  - Generate 15 sub-images of size 100-by-100
  - The length of the resulting hash is 760 bits

Rotation Modification

0 degree

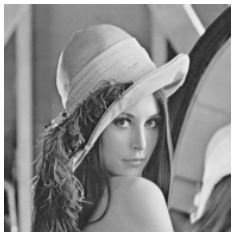
45 degree



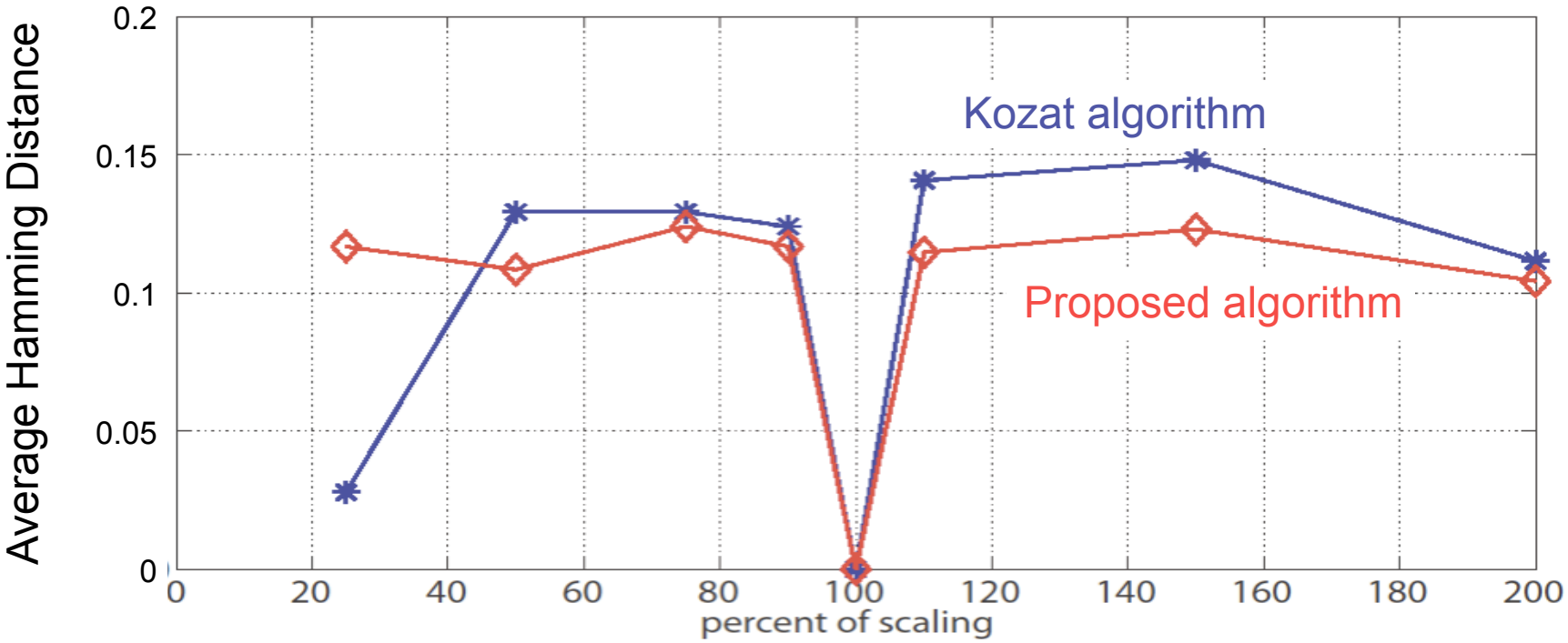
- Image normalization improves the Hamming distance under rotation modification

Scaling Modification

100% = 512 x 512



200% = 1024 x 1024



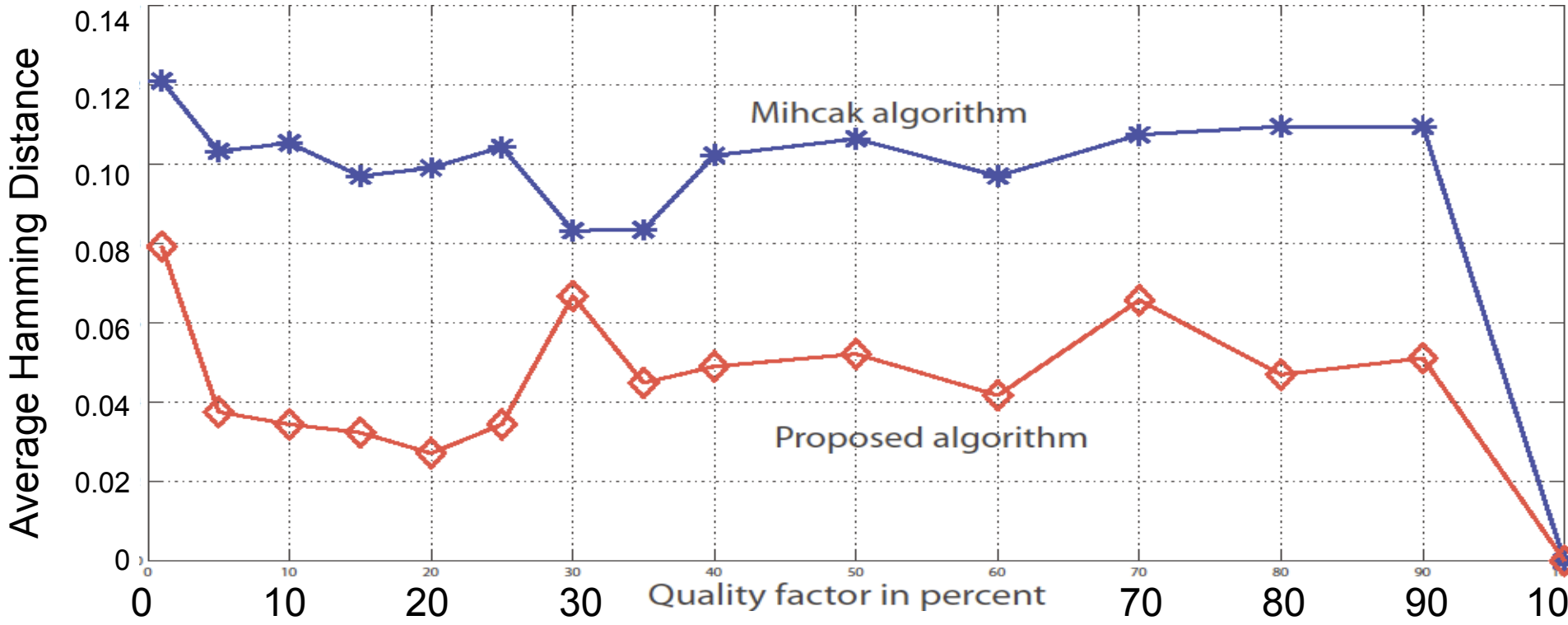
Generally better performance than Kozat et al

- Under 45% the size of sub-images is almost the size of the image.

JPEG Modification

Qf=1%

Qf=100%



- We reduce the distance to JPEG Compression.

Conclusions

- Image hashing:
  - visually similar images should produce similar hash value
  - Problem we addressed: increasing the “similarity” of the hash value
- Kozat et al image hashing based on SVD decomposition

- We improved the robustness of this algorithm
  - applied image normalization
  - significant reduction in average Hamming distance
  - against rotation, scaling and JPEG modificaitons
    - likely other affine transforms as well