Fast Indexing Strategies for Robust Image Hashes

Christian Winter, Martin Steinebach, York Yannikos

CASED Fraunhofer SIT

Outline

1. Robust Image Hashing
2. Tree-Based Index
3. LSH-Based Index
4. Evaluation of Indexing Strategies
5. Conclusion

ROBUST IMAGE HASHING

Robust Image Hashing
Use Case

Scenario

- Possession of child pornographic images forbidden
- Investigation target usually contains many thousand images to be checked
- Many people share the same illegal images

Desire of law enforcement

- Automatic classification of any content

Approach Hashing

- Automatic *recognition* of *known* illegal material
- Cryptographic hashes recognize exact copies
- Robust hashes recognize modified copies

Robust Image Hashing
ForBild hash

Idea: Block hashing

- Divide image into blocks (16×16)
- One hash bit for each block

Algorithm

- Convert image to grayscale
- Downscaling: Calculate mean brightness of each block
- Threshold: Median of block brightness values
- Set hash bit of block according to whether it is above median or not

ForBild enhancement

- Consider separate median for each quadrant
- Flipping mechanism for robustness against mirroring

Robust Image Hashing
ForBild Hash Comparison

Comparison functions

- Hamming distance: Number of non-matching bits
- Mismatch penalty:
  - Inspect non-matching bits
  - Consider distance of block brightness to median

Comparison algorithm

- Hamming distance ≤ 8 ⇨ good match
- Hamming distance > 8 and ≤ 32:
  Use mismatch penalty for decision
- Hamming distance > 32 ⇨ no match

Robust Image Hashing
ForBild Performance

Test set

- 128,036 JPEG images
- 40 GiB total size

Performance

- Hash calculation
  - 46 ms per image (average)
  - Proportional to image size
- Hash comparison
  - 5 ms per query hash
  - Proportional to database size

Extrapolation

- Advanced scenario:
  Video Hashing
- 100 M hashes in database
- Hash comparison:
  3.8 s per query hash

Implication

- Indexing strategies needed

TREE-BASED INDEX

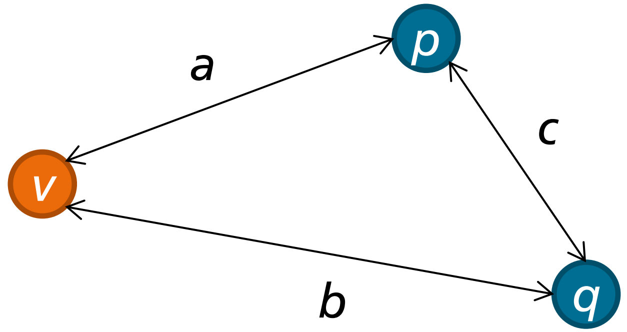
Tree-Based Index
Vantage Point trees – Introduction

Vantage point trees

- Class of metric trees
- Usage of *vantage points* for organizing data points

Triangle inequality

- Vantage point *v*
- Data points *p* and *q*
- Distances *a*, *b* and *c*
- ⇨ $c \geq | a - b |$
- Separate data points by their distance to vantage points



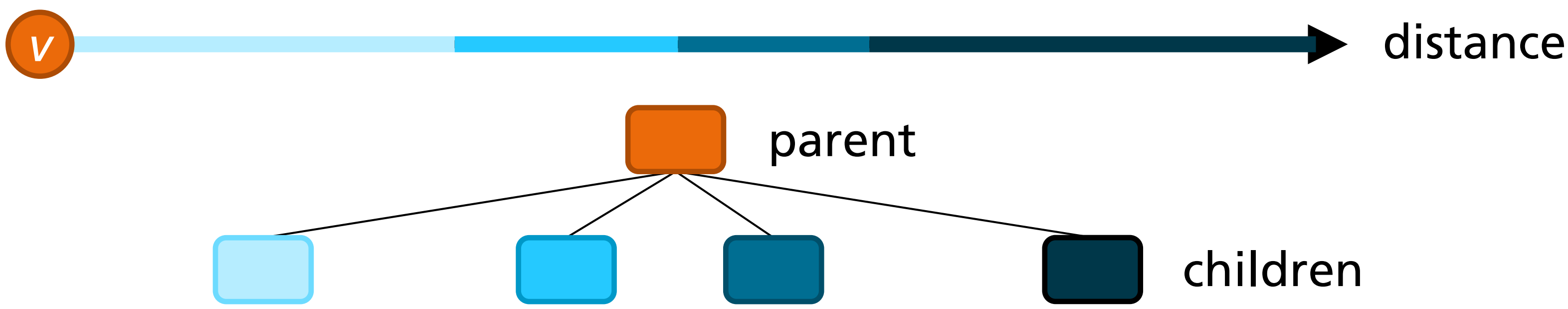
Tree-Based Index
Construction

Strategy

- One vantage point for each tree level
- Top-down construction

Algorithm

- Partition data points according to distance from vantage points
- Repeat for each level



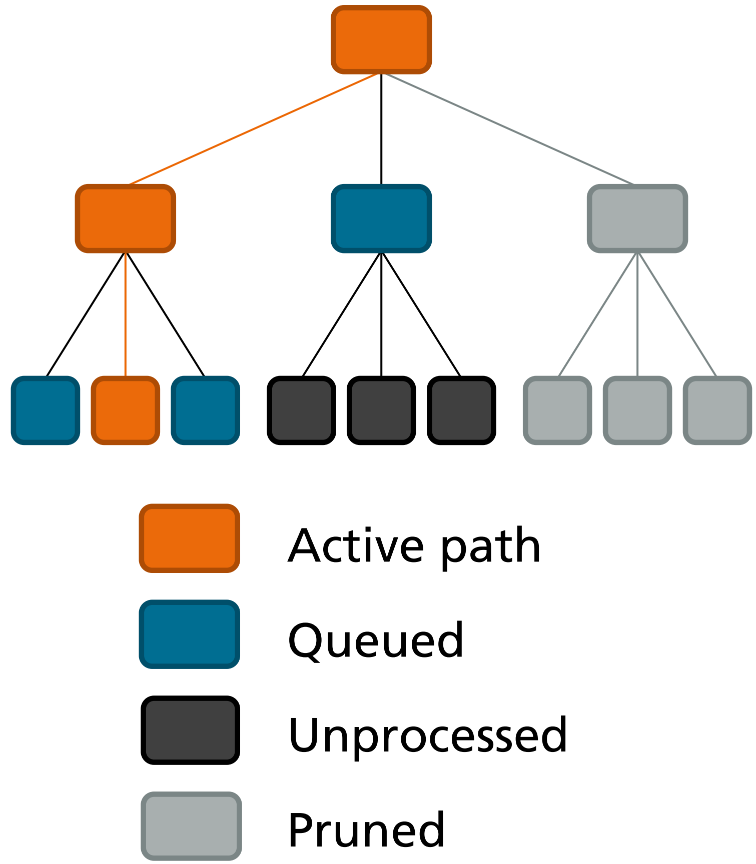
Tree-Based Index
Search Method

Query

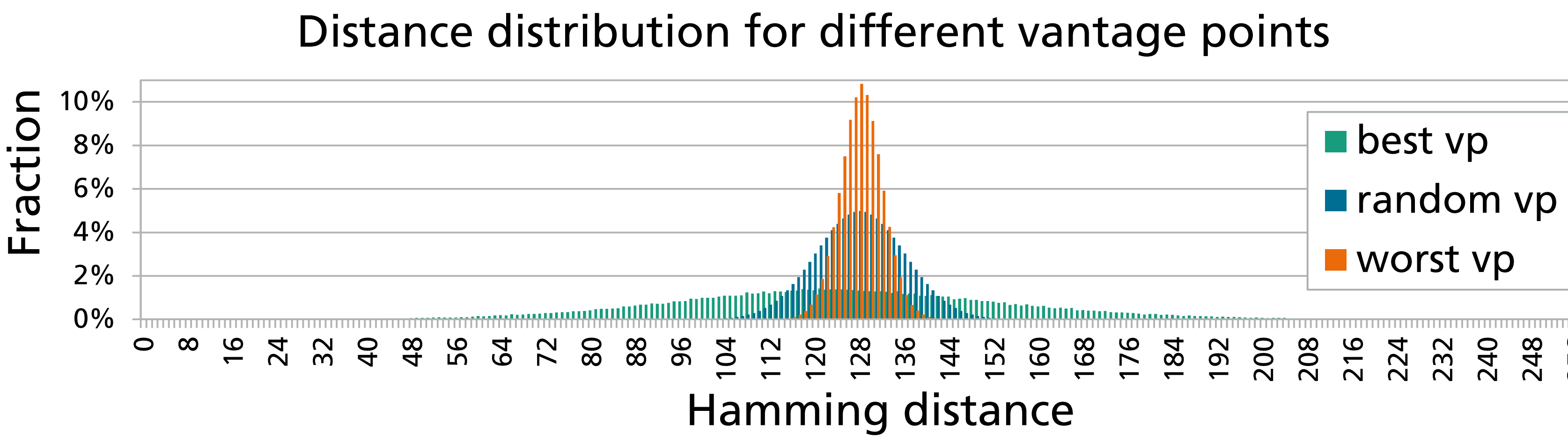
- Find closest neighbor of query point *q*
- Condition: Distance ≤ 32

Approach

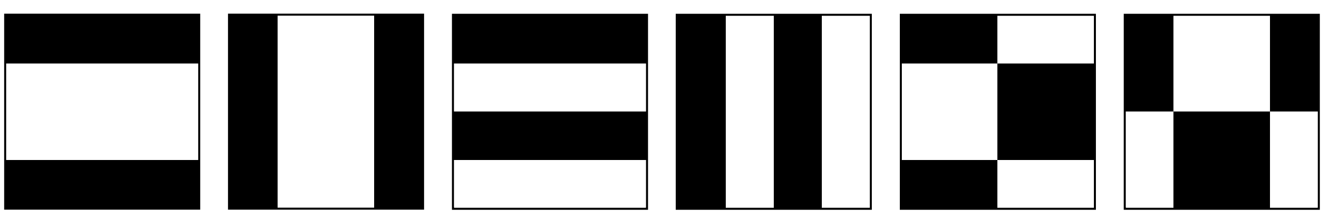
- Distances between *q* and vantage points
- Priority queue of nodes which are relevant for *q*
- Scan leafs in queue for closest neighbor



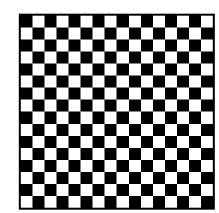
Tree-Based Index
Choice of Vantage Points



The 6 best vantage points for ForBild



Worst vp



LSH-BASED INDEX

LSH-Based Index

Locality-Sensitive Hashing (LSH)

LSH concept

- Family of hash functions (LSH family/scheme)
- Similar items hashed to same value with high probability

Bit sampling

- LSH scheme for Hamming spaces
- Each hash function selects particular bit
- Hamming distance small ⇨ great chance that selected bit matches

LSH index

- Collection of L hash tables
- Each table uses k functions from LSH family for calculating bucket address

LSH-Based Index
Configuration for ForBild

Scheme

- Bit sampling

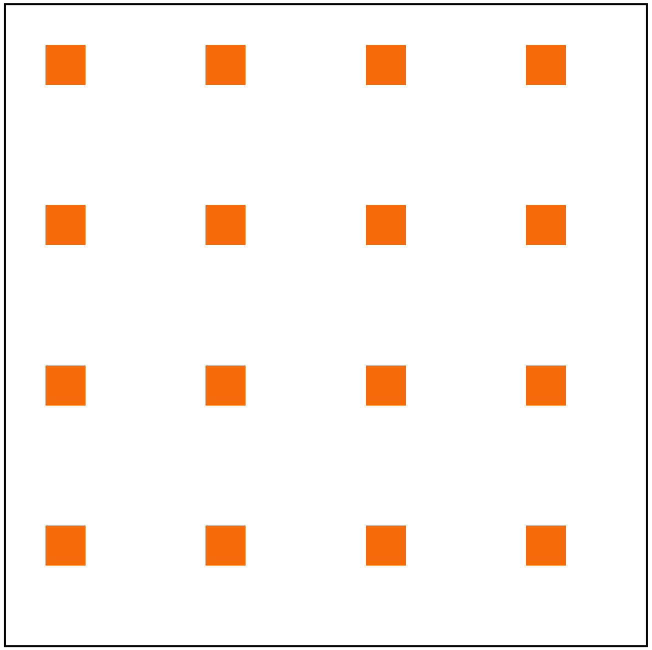
Parameters

- L = 16 tables (0…F)
- k = 16 bits per table

Bit assignment

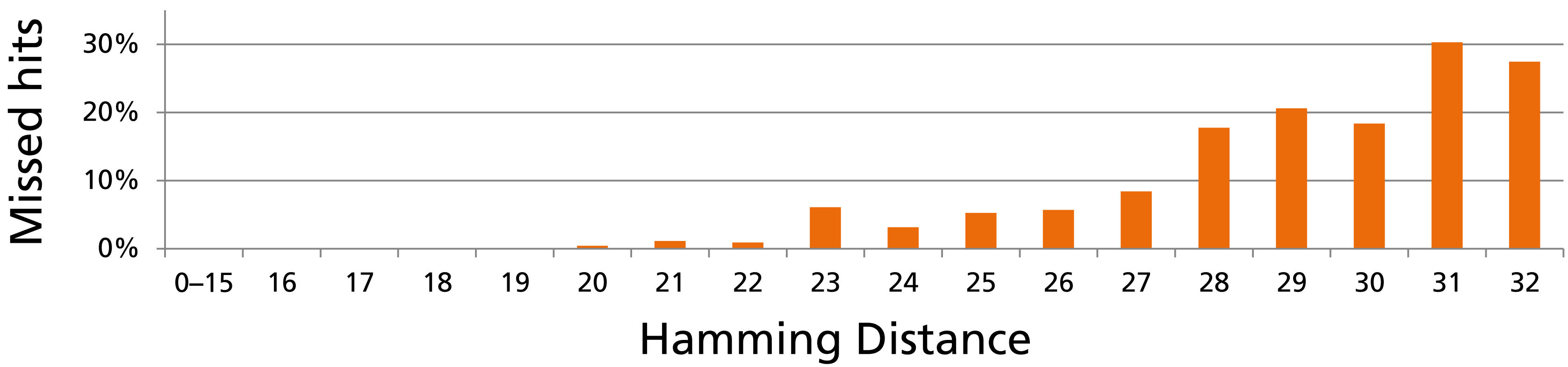
- Structured partition of the 256 bits
- Grid of 4×4 bits for each table

Example: Selected bit positions for table 5 highlighted.



LSH-Based Index
Missed Hits

- 16 tables ⇨ no missed hits for distance < 16
- Missed hits possible for distance ≥ 16
- But: Hits with large distance rare
- Empirical amount of missed hits only ≈ 0.23%
- Small number of false negatives not a problem for law enforcement



EVALUATION OF INDEXING STRATEGIES

Evaluation
Setup

Test material

- 128,036 JPEG images crawled from the Internet
- 40 GiB total size of these images
- Modified images:
  - Downscaling by 25% in each direction
  - JPEG quality 20

Hardware

- Laptop with 5 year old Intel Core2 Duo P7800
- Workstation with modern Intel Core i5-3570
  - Supports POPCNT instruction

Evaluation
Test procedure

Preparation

- Pre-calculated hashes
- Reference list
- Query list

Benchmarking task

- Search closest neighbor in reference list for each hash in query list

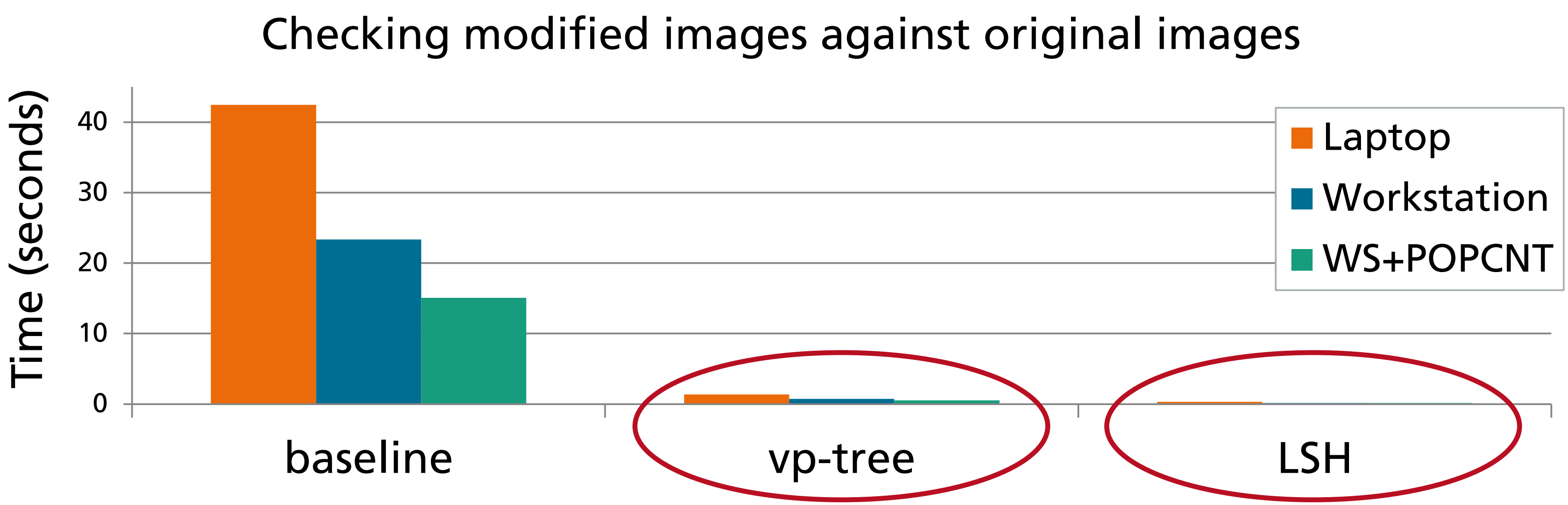
3 Algorithms

- Baseline: Efficient implementation of brute force search
- Vp-tree index
- LSH-Based index

3 Environments

- Laptop
- Workstation without POPCNT
- Workstation with POPCNT

Evaluation
Test Case 1



- Vp-tree about 30× faster than baseline
- LSH index 110–150× faster than baseline
- Hardware utilization also matters (almost factor 3)

Evaluation
Performance Comparison for Different Scenarios

Performance of vp-tree

- Query has similar match ⇨ 20–35× faster than baseline
- Query has no match ⇨ ≈ 3× faster than baseline
- Query has exact match ⇨ 100–190× faster than baseline

Performance of LSH

- Query has similar match ⇨ 110–190× faster than baseline
- Query has no match ⇨ 120–140 × faster than baseline
- Query has exact match ⇨ 110–290× faster than baseline

CONCLUSION

Conclusion

Summary

- Vantage point tree ⇨ performance boost unless image unknown
- LSH ⇨ great performance boost, but few false negatives

Bottom line

- Use vp-tree for searching thoroughly
- Use LSH for highest performance on large-scale databases

End

Thank you for your attention!

Christian Winter
christian.winter@sit.fraunhofer.de

Contact

Fraunhofer Institute for Secure Information Technology SIT

Rheinstraße 75
64295 Darmstadt
Germany

www.sit.fraunhofer.de

Christian Winter
+49 6151 869-259
christian.winter@sit.fraunhofer.de

CASED Fraunhofer SIT