Storage and Retrieval of Robotic Laser Range Data in Database Systems

Joshua Fan[1], Kaiyu Zheng[1]

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA

Joshua and Kaiyu contributed equally.

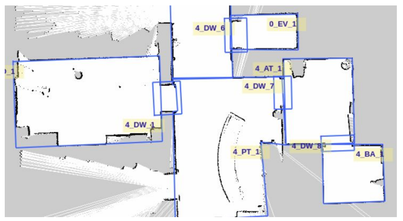
Background

A laser range-finder, commonly equipped on mobile robots, is a sensor that can detect surrounding obstacles. We have a dataset that consists of "snapshots" (200 x 200 images) of the laser range observations centered at the robot (called **"virtual scans"**). The goal of our project is to enable the **storage** of laser-range data in a database system, and the **retrieval** of similar virtual scans to a given query. Our motivation is the potential benefits in improving training semantic place classifiers in a semi-supervised learning setting.

Our problem is essentially **content-based image retrieval** for a particular type of images (virtual scans).

Dataset

Our entire dataset of virtual scans consists of 100 sequences in 3 buildings in different cities. Each sequence has between around 1500 to 3000 virtual scans. Our experiments are performed based on a subset of the entire dataset (6 sequences from 2 buildings in 2 countries).

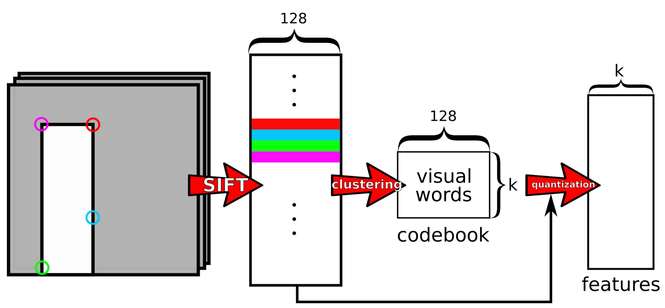


Each virtual scan is labeled by one of 10 place categories of the robot's location.

Top left: large meeting room;
Top center: doorway;
Top right: corridor;
Bottom left: portion of floor plan annotated by place categories.

Image Storage

We would like to store each image in a way that enables fast retrieval. We follow the popular Bag-of-Visual-Words model, described in (Zhou, 2017), as the representation of an image, as illustrated below.



Based on the Bag-of-Visual-Words approach, we designed a database schema with three tables: `Vscan`, `Codebook`, and `VscanFeature`. The schema is described below:

```
Vscan(id, filename, building, floor, seq, label)
Codebook(code_num, ... 128 columns of DOUBLE PRECISION)
VscanFeature(vscan_id, ... k columns of INT)
```

Retrieval Algorithms

Brute force

Simply compute distances between the query image and every image in the database (using some distance metric such as Euclidean distance), and return images with the lowest distance.

Triangle inequality pruning (Berman, 1999)

This approach aims to eliminate images using the triangle inequality. Randomly select a subset of images to be "keys", and precompute the distances between every key and each of the images in the database. (The distance metric can be customized.) These distances are stored in a new table:

```
Distances(image_id, key_id, distance)
```

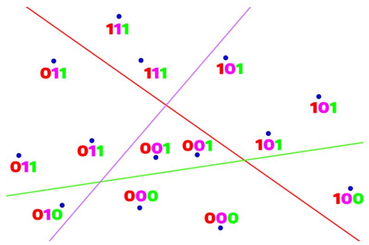
Suppose I is a database image, Q is the query image, $K_1, K_2, \ldots K_m$ are key images (arbitrary fixed images), and d is a distance metric. This lower bound on the distance $d(I, Q)$ can be derived using the Triangle Inequality:

$$d(I,Q) \geq \max_{1 \leq s \leq M} |d(I,K_s) - d(Q,K_s)|$$

Note that for any key K_s and image I, the distance $d(I, K_s)$ can be found in the table "Distances". So the only thing that has to be computed is $d(Q, K_s)$, the distance from the query image to each key. Then, you can simply use the lower bound as an estimate of the true distance (**FIDS**), or eliminate images whose lower-bound distances are above some threshold (**FIDSThreshold**).

Locality-Sensitive Hashing (Datar et al, 2004)

LSH aims to hash images into buckets such that similar images are likely to be in the same bucket. We choose random hyperplanes, and assign each image point to 0 or 1 depending on which side of the plane it is on. We combine the classifications given by each hyperplane to produce the hash, which is a binary string. Example below:



To increase the chance of finding similar images in the same hash bucket, we use multiple hash tables that are each hashed differently. We concatenate all the hash tables into as a database table:

```
HashTable(hash_table_index, hash_key, image_id)
```

We add a clustered index on (hash_table_index, hash_key), so that querying for all the images in a particular hash bucket in a particular hash table is efficient.

At retrieval time, for each hash function, we hash the query image, and add the images which hashed to the same bucket to our candidate image list to compute exact similarity. Since the image vectors were not uniformly distributed, some buckets had far more images than others, which was challenging.

References

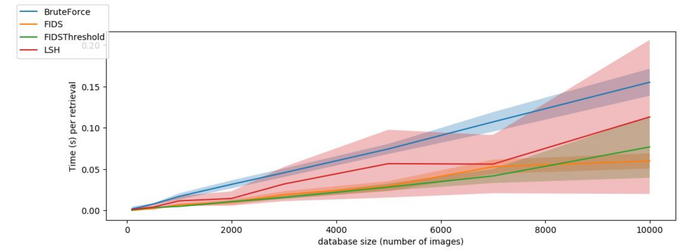
[1] Wengang Zhou, Houqiang Li, and Qi Tian. "Recent Advance in Content-based Image Retrieval: A Literature Survey". In: *arXiv preprint arXiv:1706.06064* (2017).

[2] Andrew P. Berman and Linda G. Shapiro. "A Flexible Image Database System for Content-Based Retrieval". In: *Computer Vision and Image Understanding* "75".1/2" (1999), pp. 175–195.

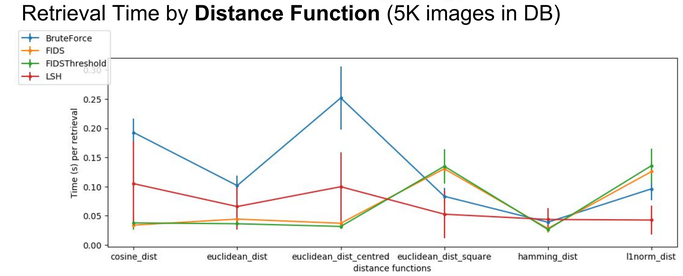
[3] Mayur Datar et al. "Locality-sensitive Hashing Scheme Based on P-stable Distributions". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Brooklyn, New York, USA: ACM, 2004, pp. 253–262.

Efficiency Results

Retrieval Time by **Scaling** (Euclidean Distance)



Retrieval Time by **Distance Function** (5K images in DB)

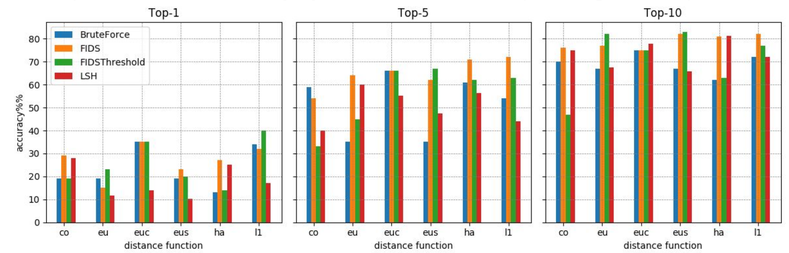


Accuracy Results

We investigated, in the top-K retrieved images, whether there **exists** one with the same label as query. In each trial, we averaged over 100 queries.

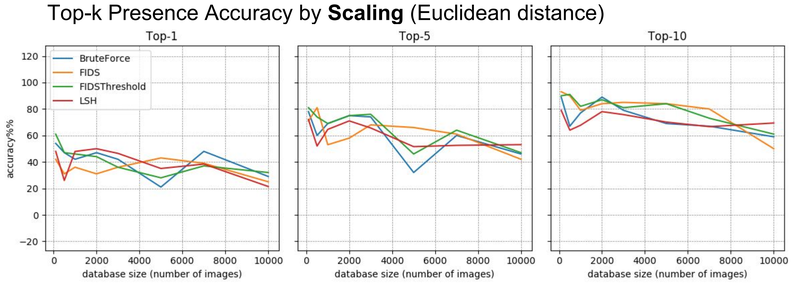
The feature table is constructed from Stockholm sequences, while query images come from Freiburg sequences.

Top-k Presence Accuracy by **Distance Function** (5K images in DB)



co = cosine distance; eu = Euclidean distance; euc = Euclidean distance centered;
eus = Euclidean distance squared; ha = Hamming distance; l1 = L1 norm distance

Top-k Presence Accuracy by **Scaling** (Euclidean distance)



Conclusion

- FIDS (approximate) is more efficient than brute force and just as accurate, even though the distance is an estimation.
  - Using exact distance isn't always better than using estimation.
- LSH also runs faster than brute force, although there is high variability due to imbalanced bucket sizes.
- Retrieval accuracy best achieved: 40% for; 75% for Top-5, and 80% for Top-10. (it is **40%-55%** for all three levels if consider majority.)
- Half of the unlabeled virtual scans can be labeled correctly, which shows potential for pre-labeling unlabeled data to improve place classifier training in semi-supervised setting.