Seminar in Robotics for CSE
"GPU-based feature matching"

Anton Maksimov[1],
supervisors: Dr. Marcin Dymczyk[2,3] and Sebastian Ratz[3]

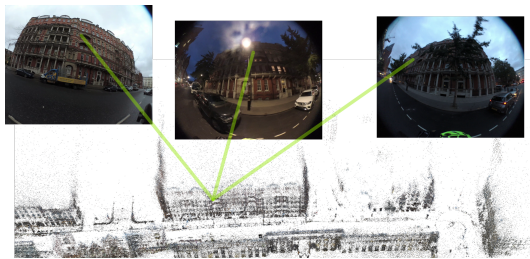
[1]CSE D-MATH, [2]ETH Zurich and [3]Sevensense Robotics AG

16.09.2020

Introduction

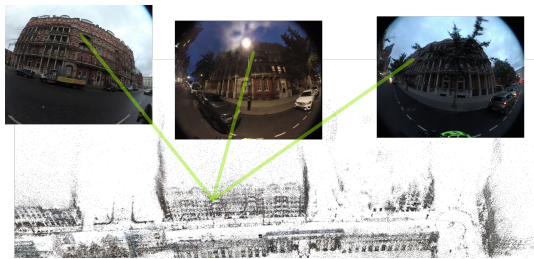
Introduction

Feature matching is one of the most fundamental techniques in computer vision: image retrieval, localization and Structure-from-Motion tasks (e.g. in `maplab` package).



Introduction

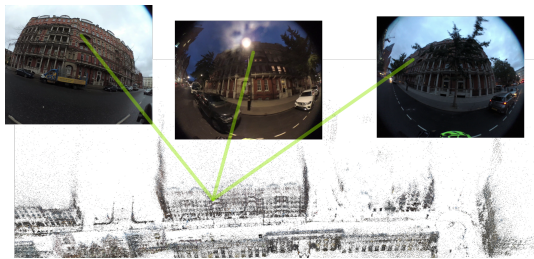
Feature matching is one of the most fundamental techniques in computer vision: image retrieval, localization and Structure-from-Motion tasks (e.g. in `maplab` package).



- Deep neural networks increased the quality of matching a lot.

Introduction

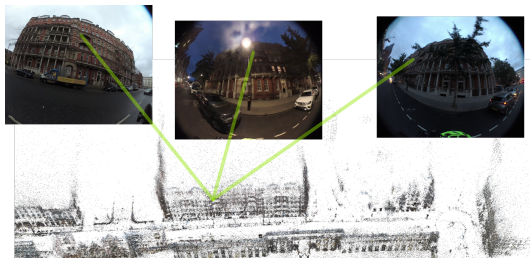
Feature matching is one of the most fundamental techniques in computer vision: image retrieval, localization and Structure-from-Motion tasks (e.g. in `maplab` package).



- Deep neural networks increased the quality of matching a lot.
- However, in some applications the crucial role plays the efficiency, latency and adaptability of the system.

Introduction

Feature matching is one of the most fundamental techniques in computer vision: image retrieval, localization and Structure-from-Motion tasks (e.g. in `maplab` package).



- Deep neural networks increased the quality of matching a lot.
- However, in some applications the crucial role plays the efficiency, latency and adaptability of the system.
- The modern rise of GPUs provides the way to achieve these goals.

Literature review

Literature review

How to obtain feature descriptors? How to measure their quality? How to match them?

Literature review

How to obtain feature descriptors? How to measure their quality? How to match them?

And specifically, how to use GPU for efficient matching?

How to obtain descriptors?

Philosophy of feature detection and description

Philosophy of feature detection and description

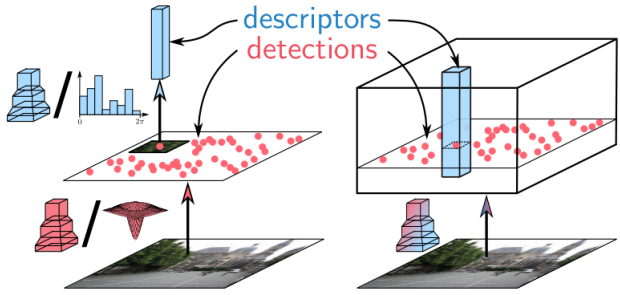
- The conventional approach – detect-then-describe $\Rightarrow$
  search for keypoints and producing of their descriptors are done sequentially and usually with hand-crafted keypoints and descriptors.

Philosophy of feature detection and description

- The conventional approach – detect-then-describe $\Rightarrow$
  search for keypoints and producing of their descriptors are done sequentially and usually with hand-crafted keypoints and descriptors.
- Novel detect-and-describe, in opposite, $\Rightarrow$
  doing these tasks simultaneously, allowing of using more information and better consistency.

Philosophy of feature detection and description

- The conventional approach – detect-then-describe ⇒
  search for keypoints and producing of their descriptors are done sequentially and usually with hand-crafted keypoints and descriptors.
- Novel detect-and-describe, in opposite, ⇒
  doing these tasks simultaneously, allowing of using more information and better consistency.

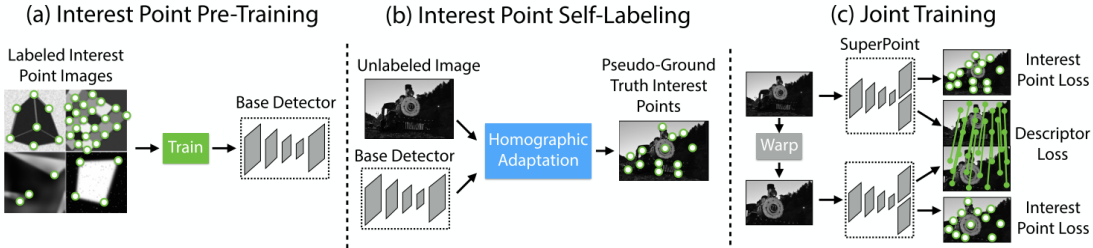


(a) detect-then-describe (b) detect-and-describe

Feature detection and description

Feature detection and description

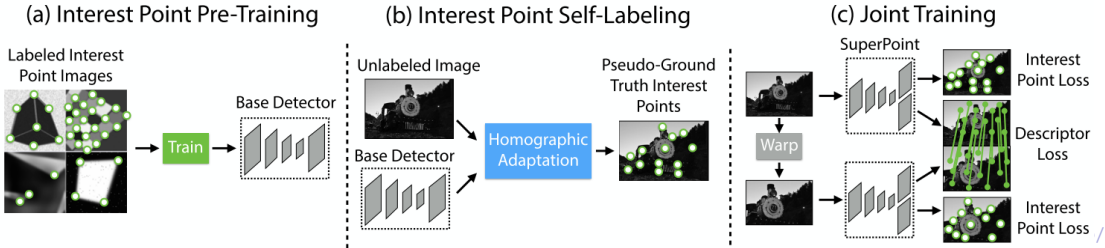
SuperPoint[1] idea: to pretrain feature detector on artificially warped geometric objects.



Feature detection and description

SuperPoint[1] idea: to pretrain feature detector on artificially warped geometric objects.

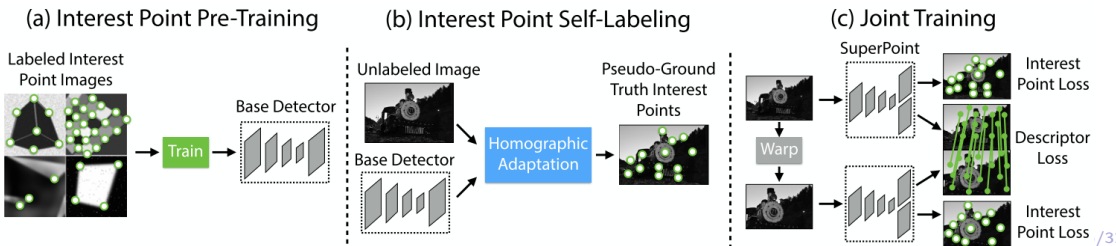
\+ Overall better quality of matching than traditional methods.



Feature detection and description

SuperPoint[1] idea: to pretrain feature detector on artificially warped geometric objects.

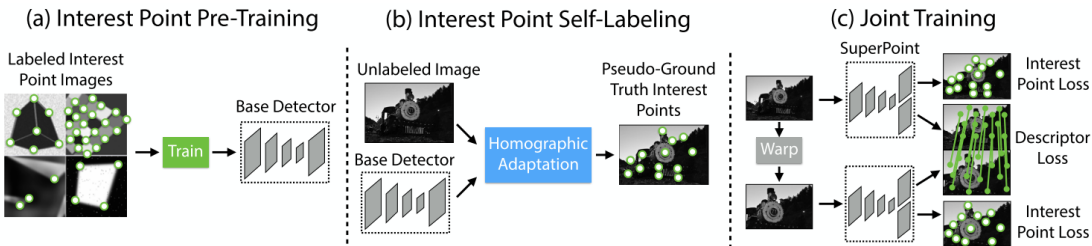
+ Overall better quality of matching than traditional methods.
+ Fully-convolutional model, needs one pass to compute descriptors.



Feature detection and description

SuperPoint[1] idea: to pretrain feature detector on artificially warped geometric objects.

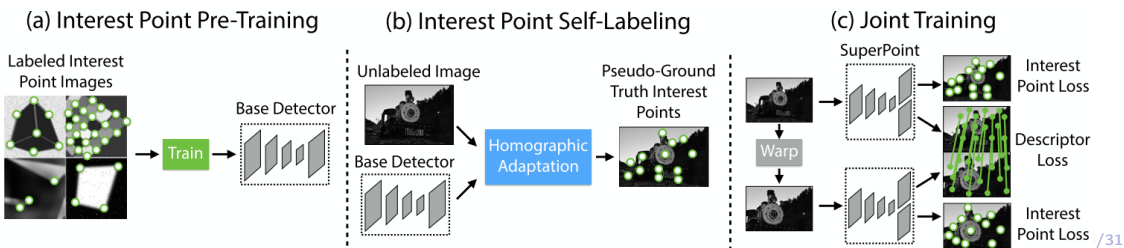
+ Overall better quality of matching than traditional methods.
+ Fully-convolutional model, needs one pass to compute descriptors.
– The descriptor isn't learnable and is fixed, it constrains its efficiency.



Feature detection and description

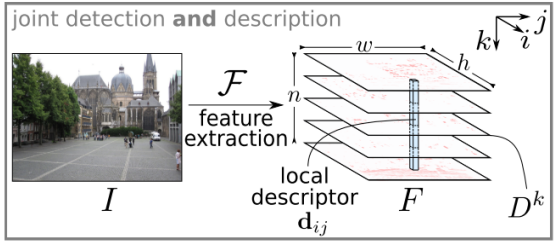
SuperPoint[1] idea: to pretrain feature detector on artificially warped geometric objects.

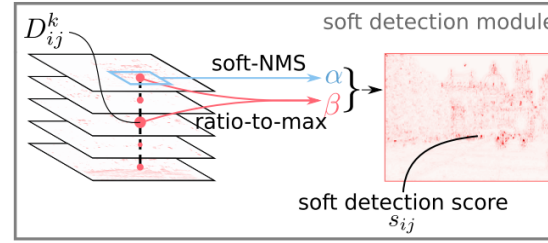
+ Overall better quality of matching than traditional methods.
+ Fully-convolutional model, needs one pass to compute descriptors.
– The descriptor isn't learnable and is fixed, it constrains its efficiency.
– Training on artificial examples introduces bias towards detecting planar geometric structures and is worse for the real world images.



Feature detection and description

D2–Net[2] idea: joint detection and description pipeline.



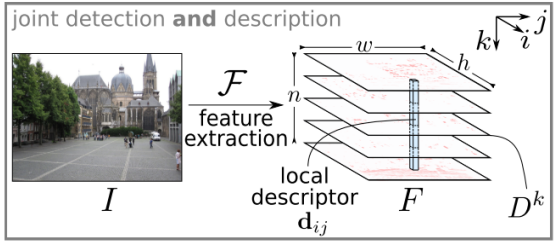


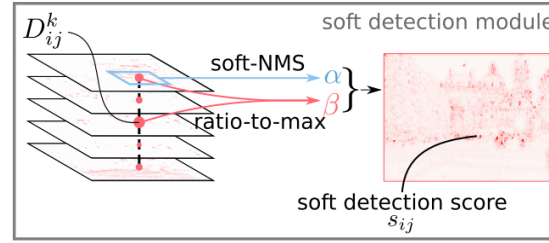
[2] Dusmanu et al., "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features", 2019.

Feature detection and description

D2–Net[2] idea: joint detection and description pipeline.

+ Was shown to be consistently better than SuperPoint and the previous ones.



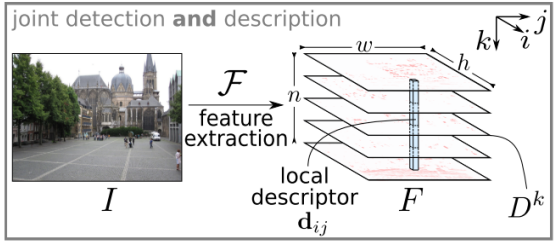


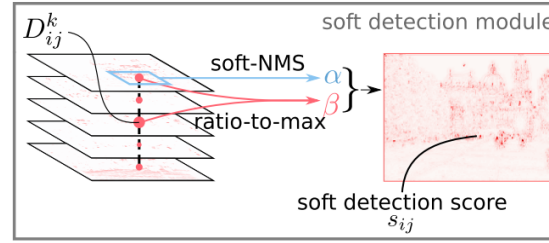
[2] Dusmanu et al., "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features", 2019.

Feature detection and description

D2–Net[2] idea: joint detection and description pipeline.

+ Was shown to be consistently better than SuperPoint and the previous ones.
– Predicts properties of keypoints in rather empiric conventional fashion (softmax, non-maximum suppression among descriptors).

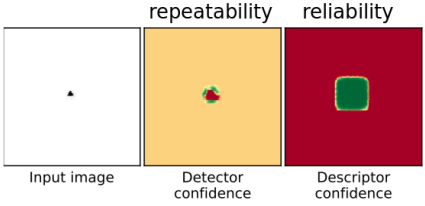


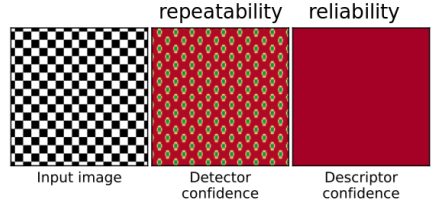


[2] Dusmanu et al., "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features", 2019.

Feature detection and description

R2D2[3] idea: estimate separately reliability and repeatability of descriptors



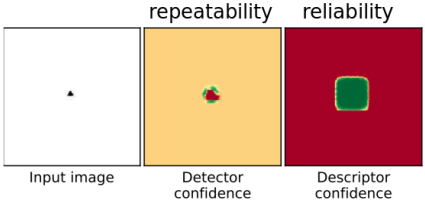


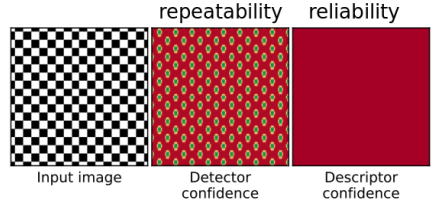
[3] Revaud et al., "R2D2: Reliable and Repeatable Detector and Descriptor", 2019.

Feature detection and description

R2D2[3] idea: estimate separately reliability and repeatability of descriptors

+ The quality of matching is at the level or better than others.



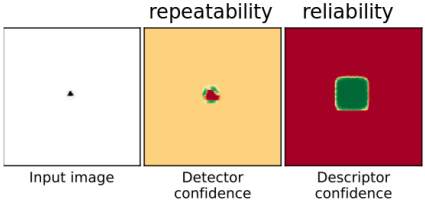


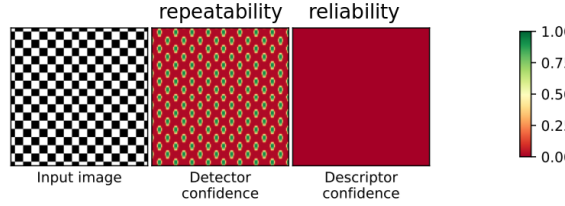
[3] Revaud et al., "R2D2: Reliable and Repeatable Detector and Descriptor", 2019.

Feature detection and description

R2D2[3] idea: estimate separately reliability and repeatability of descriptors

+ The quality of matching is at the level or better than others.
+ The model is more compact, as its descriptor length and the number of weights is much less than what the competitors have.



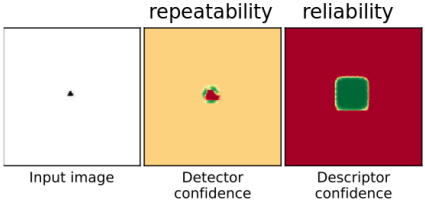


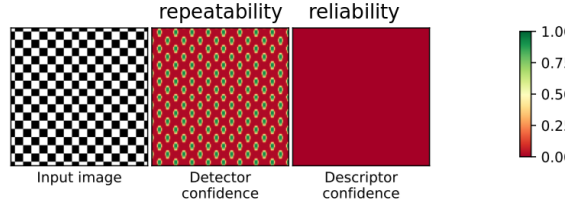
[3] Revaud et al., "R2D2: Reliable and Repeatable Detector and Descriptor", 2019.

Feature detection and description

R2D2[3] idea: estimate separately reliability and repeatability of descriptors

+ The quality of matching is at the level or better than others.
+ The model is more compact, as its descriptor length and the number of weights is much less than what the competitors have.
– Needs the choice of higher number of hyperparameters.

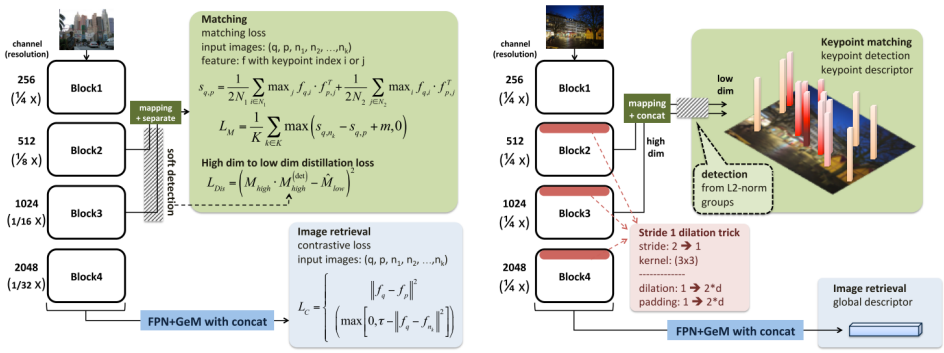




[3] Revaud et al., "R2D2: Reliable and Repeatable Detector and Descriptor", 2019.

Feature detection and description

UR2KiD[4] idea: multi-task for local (patches) and global (image retrieval) description.

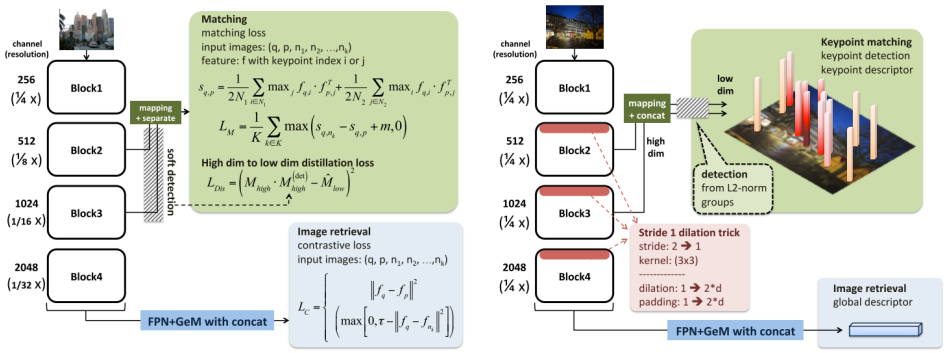


[4] Yang et al., *UR2KiD: Unifying Retrieval, Keypoint Detection, and Keypoint Description without Local Correspondence Supervision*, 2020.

Feature detection and description

UR2KiD[4] idea: multi-task for local (patches) and global (image retrieval) description.

\+ Is more robust against appearance and scale changes than the previous methods.

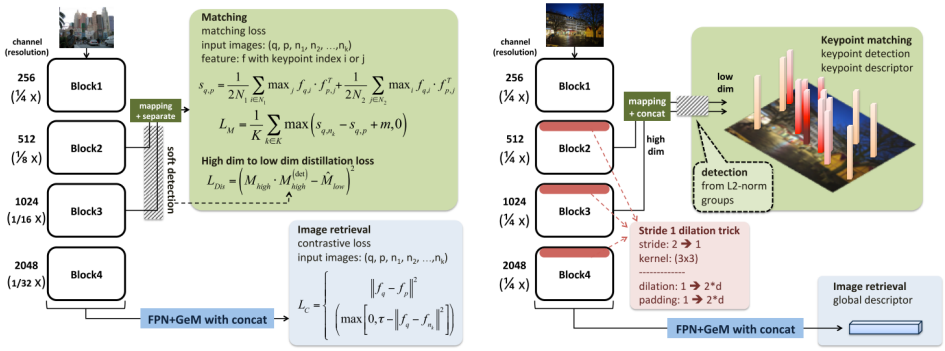


[4] Yang et al., *UR2KiD: Unifying Retrieval, Keypoint Detection, and Keypoint Description without Local Correspondence Supervision*, 2020.

Feature detection and description

UR2KiD[4] idea: multi-task for local (patches) and global (image retrieval) description.

+ Is more robust against appearance and scale changes than the previous methods.
+ Its idea of multi–task learning leads for the improvement in both tasks and could be useful in the future constructions of weakly supervised but powerful networks.



[4] Yang et al., *UR2KiD: Unifying Retrieval, Keypoint Detection, and Keypoint Description without Local Correspondence Supervision*, 2020.

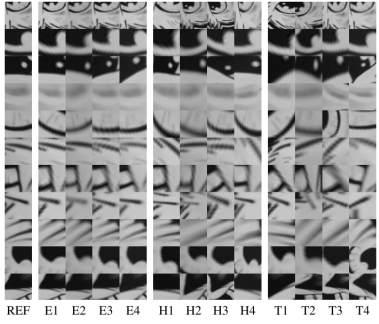
We observed descriptors construction.

We observed descriptors construction.

How to measure their quality?

Descriptors quality assessment

HPatches[5] — dataset for **2D–2D matching** assessment.
Consists of extracted and augmented with noise and transformations patches from various image sequences in varying lightning conditions and camera positions.



[5] Balntas et al., "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors", Apr. 2017.

Descriptors quality assessment

The Long–Term Visual Localization Benchmark[6] consists of several datasets of scenes under different appearance conditions for the **2D–3D** problem of estimating the 6 DoF camera pose.

[6] *The Visual Localization Benchmark, www.visuallocalization.net.*

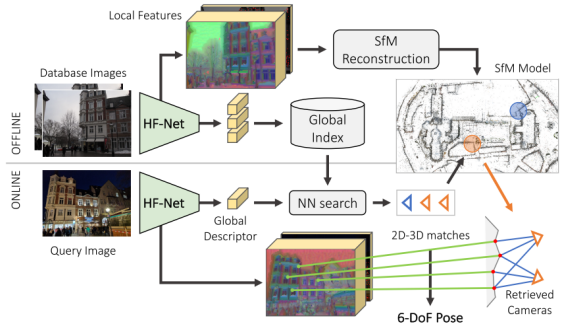
We observed how to construct good descriptors.

We observed how to construct good descriptors.

How to match them finally?

Matching of features

Hierarchical Feature Network[7] idea: coarse-to-fine pose estimation with distillation.

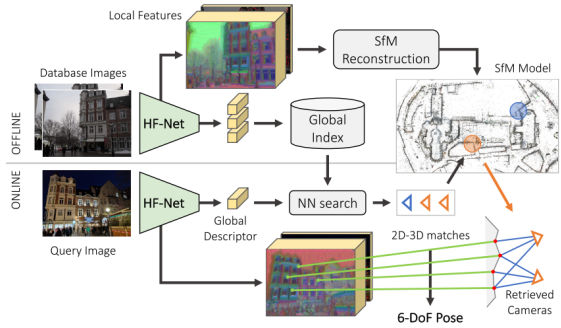


[7] Sarlin et al., "From coarse to fine: Robust hierarchical localization at large scale", 2019.

Matching of features

Hierarchical Feature Network[7] idea: coarse-to-fine pose estimation with distillation.

+ Outperformed competitors of the time on large-scale benchmarks with substantial appearance changes.

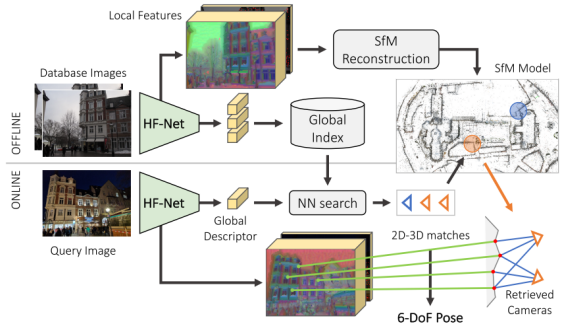


[7] Sarlin et al., "From coarse to fine: Robust hierarchical localization at large scale", 2019.

Matching of features

Hierarchical Feature Network[7] idea: coarse-to-fine pose estimation with distillation.

+ Outperformed competitors of the time on large-scale benchmarks with substantial appearance changes.
+ Distillation allows to improve the method using the global teacher signal.

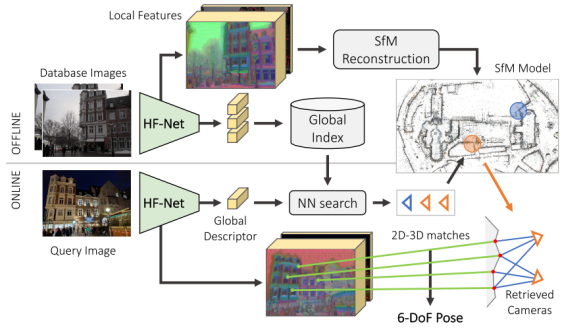


[7] Sarlin et al., "From coarse to fine: Robust hierarchical localization at large scale", 2019.

Matching of features

Hierarchical Feature Network[7] idea: coarse-to-fine pose estimation with distillation.

+ Outperformed competitors of the time on large-scale benchmarks with substantial appearance changes.
+ Distillation allows to improve the method using the global teacher signal.
+ Simultaneously estimates the camera 6-DoF pose.

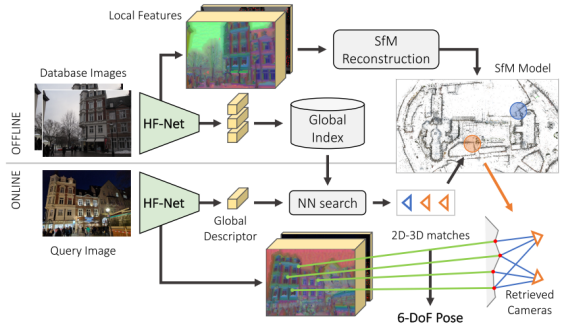


[7] Sarlin et al., "From coarse to fine: Robust hierarchical localization at large scale", 2019.

Matching of features

Hierarchical Feature Network[7] idea: coarse-to-fine pose estimation with distillation.

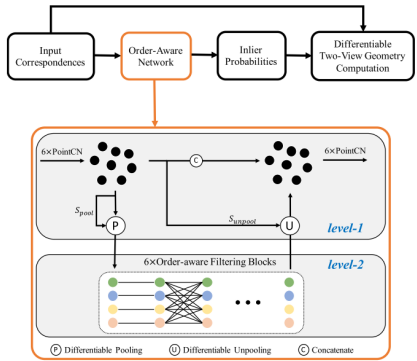
- \+ Outperformed competitors of the time on large-scale benchmarks with substantial appearance changes.
- \+ Distillation allows to improve the method using the global teacher signal.
- \+ Simultaneously estimates the camera 6-DoF pose.
- ± Tailored for mobile low computational power applications.



[7] Sarlin et al., "From coarse to fine: Robust hierarchical localization at large scale", 2019.

Matching of features

Order–aware network[8] idea: use local context and feature permutation invariance.

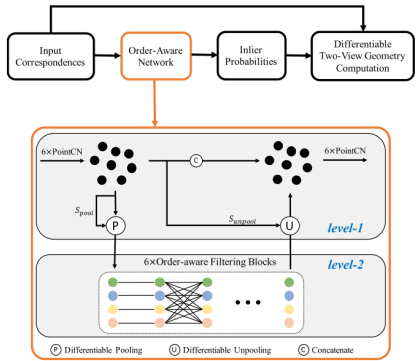


[8] Zhang et al., "Learning two-view correspondences and geometry using order-aware network", 2019.

Matching of features

Order–aware network[8] idea: use local context and feature permutation invariance.

+ Shows better performance than the other contemporary ones

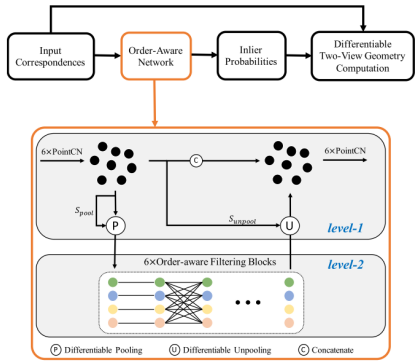


[8] Zhang et al., "Learning two-view correspondences and geometry using order-aware network", 2019.

Matching of features

Order–aware network[8] idea: use local context and feature permutation invariance.

+ Shows better performance than the other contemporary ones
+ Captures both global and local context.

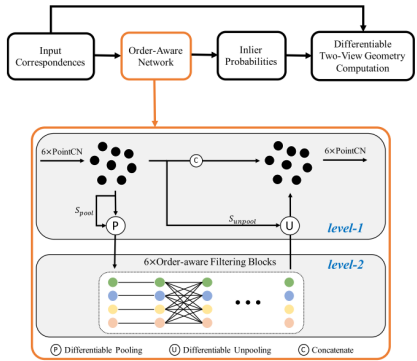


[8] Zhang et al., "Learning two-view correspondences and geometry using order-aware network", 2019.

Matching of features

Order–aware network[8] idea: use local context and feature permutation invariance.

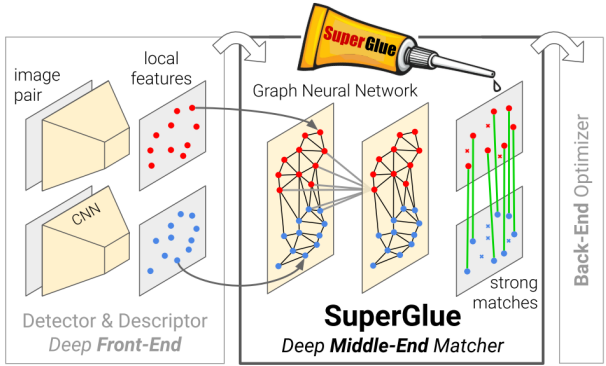
+ Shows better performance than the other contemporary ones
+ Captures both global and local context.
– Relies heavily on the quality of descriptors, as it is an outlier–rejection network uses the nearest–neighbor search.



[8] Zhang et al., "Learning two-view correspondences and geometry using order-aware network", 2019.

Matching of features

SuperGlue[9] idea: use a graph neural network and attention to solve an assignment optimization problem.

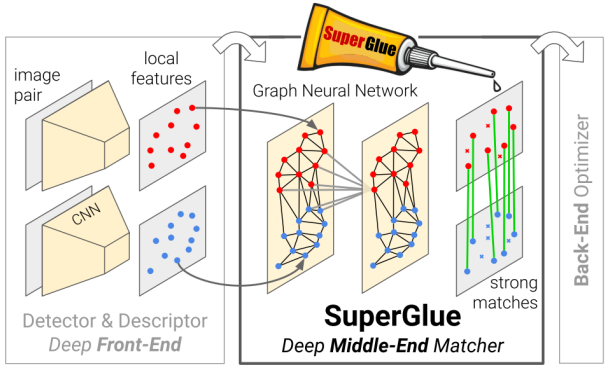


[9] Sarlin et al., "SuperGlue: Learning Feature Matching with Graph Neural Networks", 2019.

Matching of features

SuperGlue[9] idea: use a graph neural network and attention to solve an assignment optimization problem.

+ Achieves significant improvement over existing approaches, working real–time and well both with learned and traditional features.



[9] Sarlin et al., "SuperGlue: Learning Feature Matching with Graph Neural Networks", 2019.

How to leverage GPU for more efficient matching?
(e.g. with huge databases)

GPU-based approaches

Nagy et al.[10] describe GPU–based **acceleration for visual–inertial odometry**.

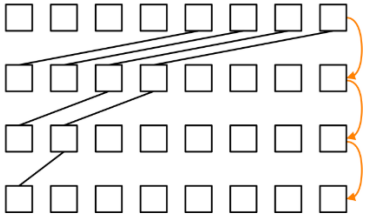


Fig. 2. Warp-level communication pattern during cell maximum selection. At the end of the communication, thread 0 has the valid maximum.

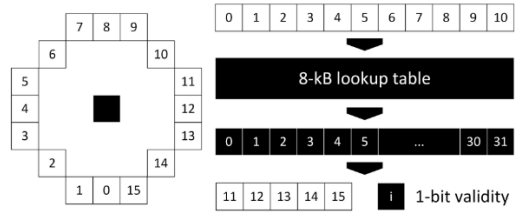


Fig. 3. FAST corner point evaluation with an 8 kilobyte lookup table

[10] Nagy, Foehn, and Scaramuzza, *Faster than FAST: GPU-Accelerated Frontend for High-Speed VIO*, 2020.

GPU-based approaches

Nagy et al.[10] describe GPU–based **acceleration for visual–inertial odometry**.
Idea: optimize on the low–level the most prominent for latency operations:
non–maximum suppression and consequent feature selection.

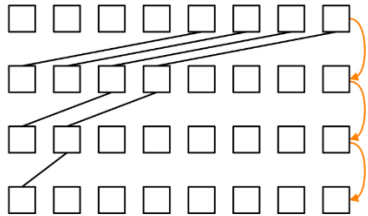


Fig. 2. Warp-level communication pattern during cell maximum selection. At the end of the communication, thread 0 has the valid maximum.

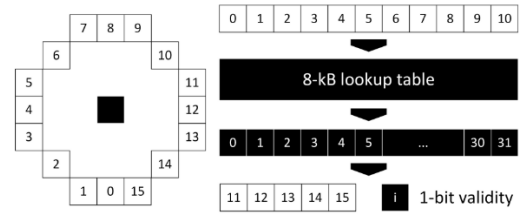


Fig. 3. FAST corner point evaluation with an 8 kilobyte lookup table

[10] Nagy, Foehn, and Scaramuzza, *Faster than FAST: GPU-Accelerated Frontend for High-Speed VIO*, 2020.

GPU-based approaches

Nagy et al.[10] describe GPU–based **acceleration for visual–inertial odometry**.
Idea: optimize on the low–level the most prominent for latency operations:
non–maximum suppression and consequent feature selection.

+ Much faster than competitors.

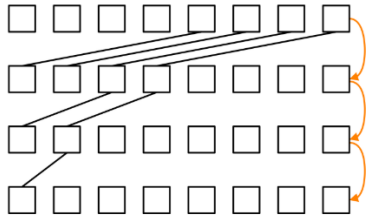


Fig. 2. Warp-level communication pattern during cell maximum selection. At the end of the communication, thread 0 has the valid maximum.

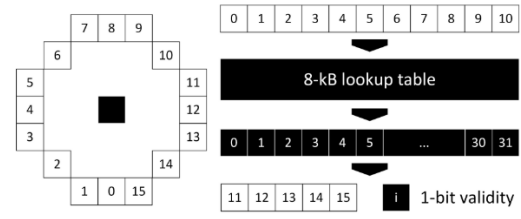


Fig. 3. FAST corner point evaluation with an 8 kilobyte lookup table

[10] Nagy, Foehn, and Scaramuzza, *Faster than FAST: GPU-Accelerated Frontend for High-Speed VIO*, 2020.

GPU-based approaches

Nagy et al.[10] describe GPU–based **acceleration for visual–inertial odometry**.
Idea: optimize on the low–level the most prominent for latency operations:
non–maximum suppression and consequent feature selection.

+ Much faster than competitors.
± Uniform distribution of features.

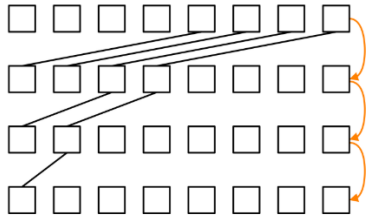


Fig. 2. Warp-level communication pattern during cell maximum selection. At the end of the communication, thread 0 has the valid maximum.

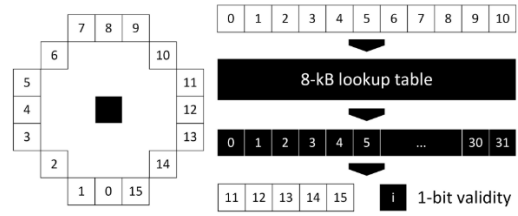
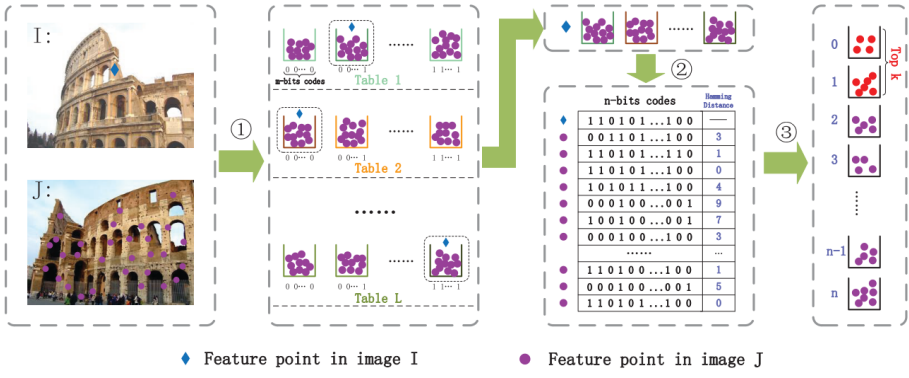


Fig. 3. FAST corner point evaluation with an 8 kilobyte lookup table

[10] Nagy, Foehn, and Scaramuzza, *Faster than FAST: GPU-Accelerated Frontend for High-Speed VIO*, 2020.

GPU-based approaches

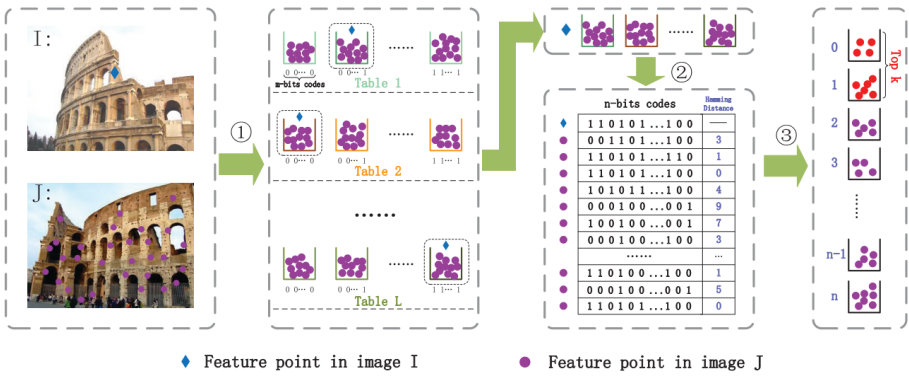
Cascade Hashing[11] idea: reduce number of candidates by hierarchical space hashing.



[11] Cheng et al., "Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction", 2014.

GPU-based approaches

Cascade Hashing[11] idea: reduce number of candidates by hierarchical space hashing. Dissects space multiple times with random planes and binary encodes descriptors by containing in one or another half-spaces.

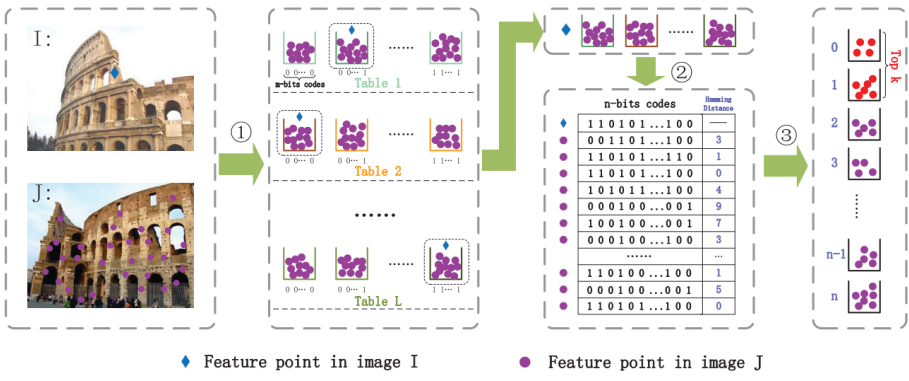


[11] Cheng et al., "Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction", 2014.

GPU-based approaches

Cascade Hashing[11] idea: reduce number of candidates by hierarchical space hashing. Dissects space multiple times with random planes and binary encodes descriptors by containing in one or another half-spaces.

+ Much faster than competitors.

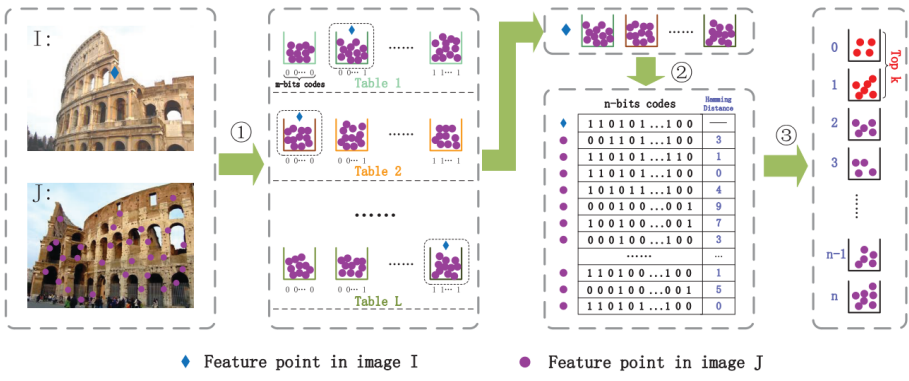


[11] Cheng et al., "Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction", 2014.

GPU-based approaches

Cascade Hashing[11] idea: reduce number of candidates by hierarchical space hashing. Dissects space multiple times with random planes and binary encodes descriptors by containing in one or another half-spaces.

+ Much faster than competitors.
+ Robust to noise, unsupervised.



[11] Cheng et al., "Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction", 2014.

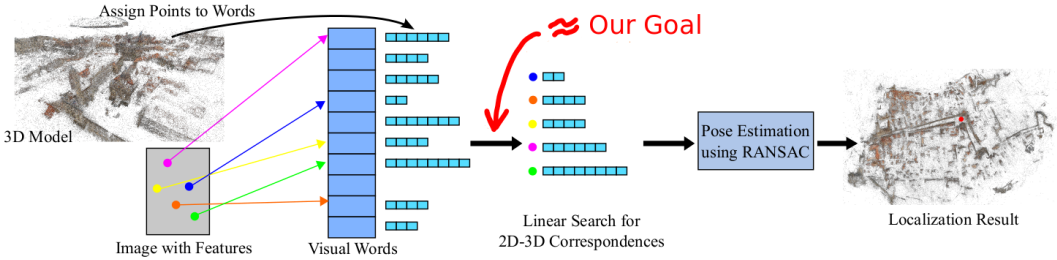
Literature $\Rightarrow$ experience in reality

Literature $\Rightarrow$ experience in reality

General problem: matching descriptors from 2D image to 3D database or between two images. This is the fundamental operation in computer vision.

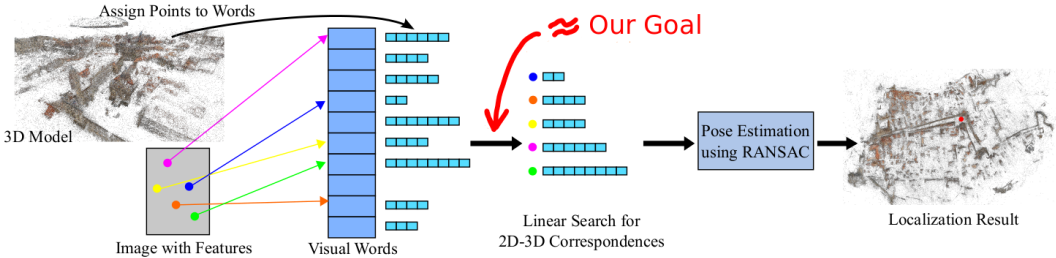
Literature ⇒ experience in reality

General problem: matching descriptors from 2D image to 3D database or between two images. This is the fundamental operation in computer vision.



Literature ⇒ experience in reality

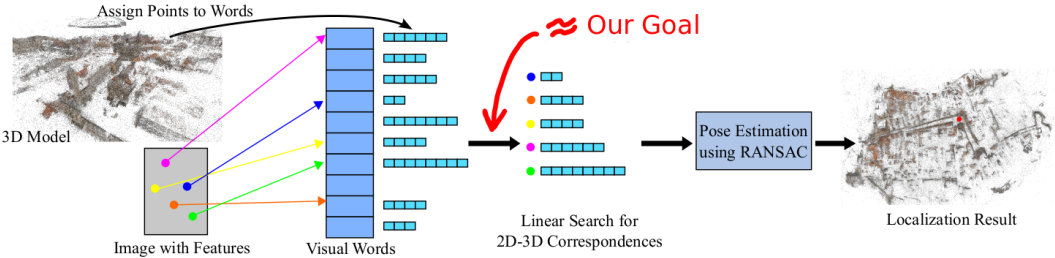
General problem: matching descriptors from 2D image to 3D database or between two images. This is the fundamental operation in computer vision.



Idea:
- to try different methods for finding the nearest–neighbor for descriptors,
- compare CPU and GPU versions.

Literature ⇒ experience in reality

General problem: matching descriptors from 2D image to 3D database or between two images. This is the fundamental operation in computer vision.

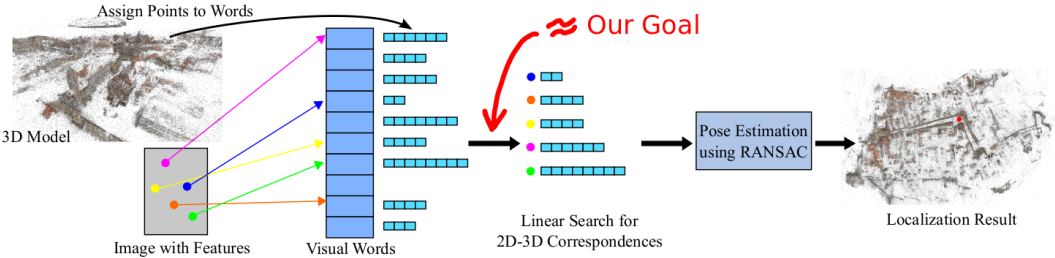


Idea: • to try different methods for finding the nearest–neighbor for descriptors,
• compare CPU and GPU versions.

Formulation: we have $N \in \mathbb{R}^F$ query descriptors and $N_{\text{db}} \in \mathbb{R}^F$ database descriptors.

Literature ⇒ experience in reality

General problem: matching descriptors from 2D image to 3D database or between two images. This is the fundamental operation in computer vision.



Idea: • to try different methods for finding the nearest–neighbor for descriptors,
• compare CPU and GPU versions.

Formulation: we have $N \in \mathbb{R}^F$ query descriptors and $N_{\text{db}} \in \mathbb{R}^F$ database descriptors.

Goal: find the closest descriptor in DB (in Euclidean metric) for each one in the query.

Methods compared

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.
2. Further methods use encodings $\mathbf{h}_i$ of descriptors as a binned distances to the set of random planes.

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.
2. Further methods use encodings $\mathbf{h}_i$ of descriptors as a binned distances to the set of random planes.

$$h_{i,j} = \left\lfloor \frac{\mathbf{a}_j \cdot \mathbf{d}_i + \mathbf{b}_j}{W} \right\rfloor, \quad \text{where } \mathbf{a}_j \sim \mathcal{N}(0,1)^{M \times F},\ \mathbf{b}_j \sim \text{Unif}(0, W)$$

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.

2. Further methods use encodings $\mathbf{h}_i$ of descriptors as a binned distances to the set of random planes.

$$h_{i,j} = \left\lfloor \frac{\mathbf{a}_j \cdot \mathbf{d}_i + \mathbf{b}_j}{W} \right\rfloor, \quad \text{where } \mathbf{a}_j \sim \mathcal{N}(0,1)^{M \times F},\ \mathbf{b}_j \sim \text{Unif}(0, W)$$

   2.1 "Manhattan check in encoded space": candidates to check are chosen by the **closest encodings in Manhattan distance**.

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.

2. Further methods use encodings $\mathbf{h}_i$ of descriptors as a binned distances to the set of random planes.

$$h_{i,j} = \left\lfloor \frac{\mathbf{a}_j \cdot \mathbf{d}_i + \mathbf{b}_j}{W} \right\rfloor, \quad \text{where } \mathbf{a}_j \sim \mathcal{N}(0,1)^{M \times F},\ \mathbf{b}_j \sim \text{Unif}(0, W)$$

   2.1 "Manhattan check in encoded space": candidates to check are chosen by the **closest encodings in Manhattan distance**.

   2.2 Locally Sensitive Hashing (LSH)[12]: obtained **binned encodings are hashed** using polynomial hashing several times. **For faster search of hits linked lists (chaining)** of DB occurrences by their value are used.

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Methods compared

1. Brute–force: **check all distances** ($N \cdot N_{\text{db}}$) and find the closest to each of the query descriptors among the DB ones.

2. Further methods use encodings $\mathbf{h}_i$ of descriptors as a binned distances to the set of random planes.

$$h_{i,j} = \left\lfloor \frac{\mathbf{a}_j \cdot \mathbf{d}_i + \mathbf{b}_j}{W} \right\rfloor, \quad \text{where } \mathbf{a}_j \sim \mathcal{N}(0,1)^{M \times F},\ \mathbf{b}_j \sim \text{Unif}(0, W)$$

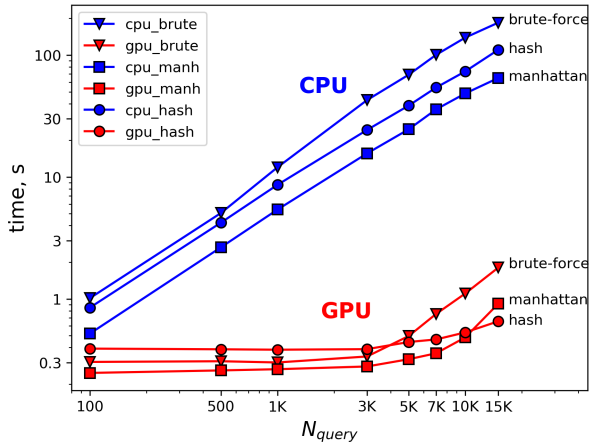
   2.1 "Manhattan check in encoded space": candidates to check are chosen by the **closest encodings in Manhattan distance**.

   2.2 Locally Sensitive Hashing (LSH)[12]: obtained **binned encodings are hashed** using polynomial hashing several times. **For faster search of hits linked lists (chaining)** of DB occurrences by their value are used.

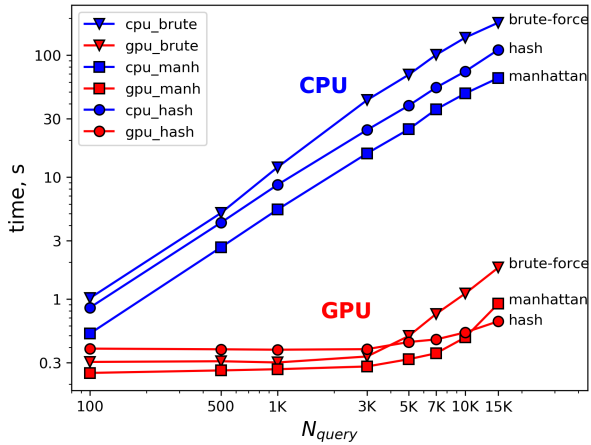
   For these two methods we use empirically the most reasonable parameters.

[12] Shakhnarovich, Darrell, and Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)", 2005.

Time depending on algorithm, machine and **query** size ($N_{\text{db}} \approx 28\text{K}$)

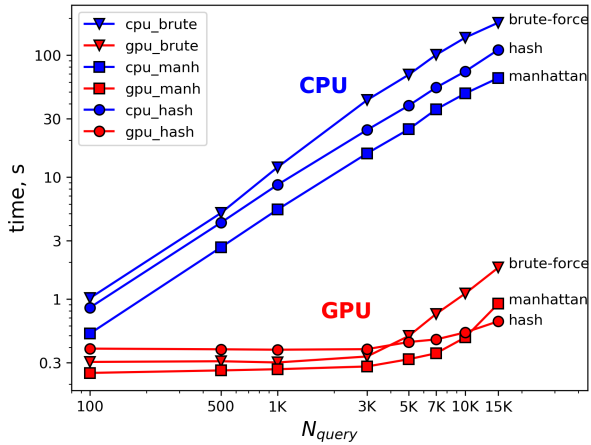


Time depending on algorithm, machine and **query** size ($N_{\text{db}} \approx 28\text{K}$)



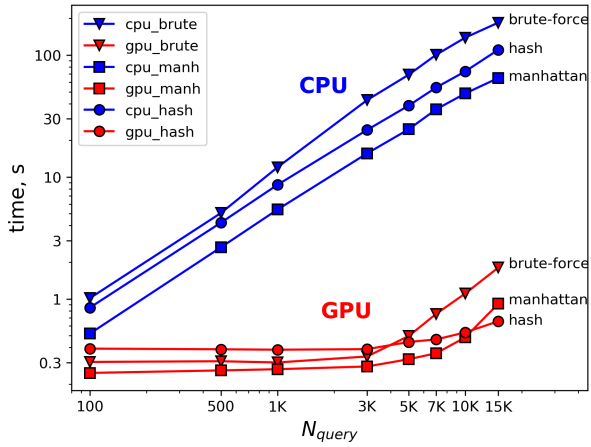
- GPU is much faster than CPU, esp. when big query – 2 orders change.

Time depending on algorithm, machine and **query** size ($N_{db} \approx 28K$)



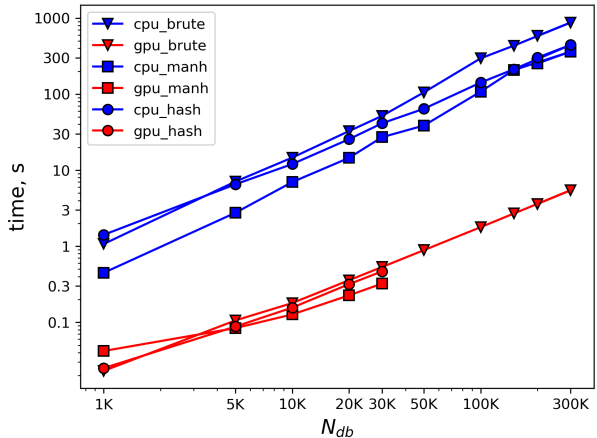
- GPU is much faster than CPU, esp. when big query – 2 orders change.
- Difference between algorithms is not so big, it depends on parameters.

Time depending on algorithm, machine and **query** size ($N_{\text{db}} \approx 28\text{K}$)

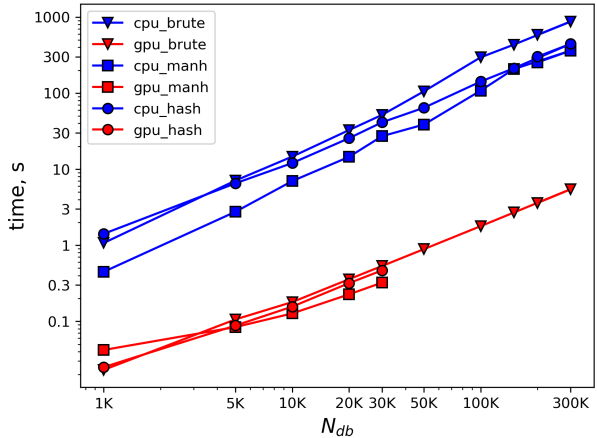


- GPU is much faster than CPU, esp. when big query – 2 orders change.
- Difference between algorithms is not so big, it depends on parameters.
- Possible explanation for blow-up of GPU with big queries - GPU memory overload.

Sweep over different DB sizes with constant query ($N_{\text{query}} = 5\text{K}$)



Sweep over different DB sizes with constant query ($N_{\text{query}} = 5\text{K}$)



- Perfect linear time dependency for both CPU and GPU, as expected.

Resume

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.

gpu_brute, 30 closest correspondences

gpu_hash, 30 closest correspondences

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.
- Possible low-level tweaks could speed it up potentially.

gpu_brute, 30 closest correspondences

gpu_hash, 30 closest correspondences

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.
- Possible low-level tweaks could speed it up potentially.
- Hashing is very flexible with several parameters to tune.

gpu_brute, 30 closest correspondences

gpu_hash, 30 closest correspondences

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.
- Possible low-level tweaks could speed it up potentially.
- Hashing is very flexible with several parameters to tune.
- Hashing could be useful when speed is more important than quality or memory constraints exist.

gpu_brute, 30 closest correspondences

gpu_hash, 30 closest correspondences

Resume

- GPU allows to speed up the CPU matching by up to 2 orders magnitude.
- Hashing in our experiments can't achieve both better speed and the very same quality of matching as GPU brute-force.
- Possible low-level tweaks could speed it up potentially.
- Hashing is very flexible with several parameters to tune.
- Hashing could be useful when speed is more important than quality or memory constraints exist.
- Parameter tuning is not obvious for hashing.

gpu_brute, 30 closest correspondences

gpu_hash, 30 closest correspondences

Questions?

gpu_brute, 30 closest correspondences

gpu_manh, 30 closest correspondences

gpu_hash, 30 closest correspondences

How to find the best parameters? Which quality metric to use?

Metric of quality is the average Euclidean distance ("dist") in corresponding pairs.

t, s	dist	Parameters			
		L	tableSize	M	W
0.496	**0.596**	ref. GPU brute–force			
0.492	0.602	3	17	1	0.5
0.443	0.624	3	37	1	0.1
0.254	0.654	1	83	3	0.5
0.172	0.678	1	83	3	0.1

Table 1: Example of dependence of results on the parameters of the method, for LSH

Results for $N_{\text{query}} = 5000, N_{\text{db}} = 28188$

Type	Mode	Av. dist	t, s	std(t), s	std(dist)	CPU/GPU
Brute-force	CPU	0.595	46.133	0.023	0	~ 80
	GPU		0.496	0.003	0	
Manhattan	CPU	0.608	24.632	2.585	0.066	
	GPU		0.32	0.01	0.066	
LS Hashing	CPU	0.623	38.562	2.077	0.066	
	GPU		0.439	0.006	0.066	

Table 2: Comparison of methods with the empirically best parameters

"Best" taken parameters:
for Manhattan: $M = 1, W = 1$,
for LSH: $L = 3, tableSize = 83, M = 1, W = 0.1$.

References

- Balntas, Vassileios et al. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (Apr. 2017). DOI: 10.1109/TPAMI.2019.2915233.
- Cheng, J. et al. "Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1–8.
- DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2018), pp. 337–33712.
- Dusmanu, Mihai et al. "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features". In: *ArXiv* abs/1905.03561 (2019).
- Nagy, Balazs, Philipp Foehn, and Davide Scaramuzza. *Faster than FAST: GPU-Accelerated Frontend for High-Speed VIO*. 2020. arXiv: 2003.13493 [cs.CV].

References (cont.)

- Revaud, Jerome et al. "R2D2: Reliable and Repeatable Detector and Descriptor". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 12405–12415. URL: http://papers.nips.cc/paper/9407-r2d2-reliable-and-repeatable-detector-and-descriptor.pdf.
- Sarlin, Paul Edouard et al. "From coarse to fine: Robust hierarchical localization at large scale". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (2019), pp. 12708–12717. ISSN: 10636919. DOI: 10.1109/CVPR.2019.01300. arXiv: 1812.03506.
- Sarlin, Paul-Edouard et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: (2019). arXiv: 1911.11763. URL: http://arxiv.org/abs/1911.11763.
- Shakhnarovich, Gregory, Trevor Darrell, and P. Indyk. "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)". In: 2005.
- *The Visual Localization Benchmark, www.visuallocalization.net*. https://www.visuallocalization.net/.

References (cont.)

- Yang, Tsun-Yi et al. *UR2KiD: Unifying Retrieval, Keypoint Detection, and Keypoint Description without Local Correspondence Supervision*. 2020. arXiv: 2001.07252 [cs.CV].
- Zhang, Jiahui et al. "Learning two-view correspondences and geometry using order-aware network". In: *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob (2019), pp. 5844–5853. ISSN: 15505499. DOI: 10.1109/ICCV.2019.00594. arXiv: 1908.04964.