Seminar in Robotics for CSE

"GPU-based feature matching".

Student:	Anton Maksimov	Start Date:	21 Apr 2020
Advisors:	Marcin Dymczyk	End Date:	18 Sep 2020
	Sebastian Ratz		

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

GPU-based feature matching

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Maksimov	Anton

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 18.09.2020	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Introduction

Feature matching is one of the most **fundamental techniques in computer vision**. It is crucial for image retrieval, localization and Structure–from–Motion tasks.
The recent success of deep neural networks increased the quality of matching to the unprecedented level.
However, in some applications, such as autonomous cars, drones, virtual and augmented reality, the crucial role plays the **efficiency, latency and adaptability of the system**.
The modern rise of GPUs provides the way to achieve these goals.

Related Work

Descriptors quality assessment

To assess the quality of descriptors we need to have some dataset of images and evaluation metric.

Evaluation of the quality of descriptors for the local 2D–2D matching could be done using dataset **HPatches** [1], which consists of extracted and augmented with noise and transformations patches from various image sequences in varying lightning conditions and camera positions. Assessment metric is the corresponding Average Precision. Three different use cases are assessed: patch verification, image matching, and patch retrieval. This evaluation method was shown to be more reliable than the previous ones.

The Long–Term Visual Localization Benchmark [2] consists of several datasets for the 2D–3D problem of estimating the 6 DoF camera pose from which a given image was taken relative to a reference scene representation under different appearance conditions. The metric is the percentage of localized poses with small error relatively to the true pose. It is thresholded with maximal distance and rotation to be counted: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°)

Feature detection and description

Traditional interest point detectors rely on human guess of how it is better to find parts of image which are consistent among different positions of the objects and various lightning conditions. Examples of this approach are SIFT [3] and its computationally more efficient successors SURF [4] and BRISK [5].

However, including this task in the whole image matching pipeline could provide end–to–end matching using single neural network, which becomes self–supervised and therefore potentially more efficient, as it introduces less human bias and allows for its parts to adapt to each other.

This approach is called *detect–and–describe*, meaning simultaneous search for keypoints and their descriptors in one network, as opposite to the conventional *detect–then–describe* approach, in which these tasks are done sequentially and usually with hand–crafted keypoints and descriptors.

Among the examples of the *detect–and–describe* approach are

- **SuperPoint** [6] — its main idea is to use detector subnetwork pretrained on synthetic data. This **training data** is of a simple polygonal–type with thus known positions of keypoints. It is also augmented with different not too extreme homographies, the same is done with test images. Real image **training data** is from MS–COCO [7] dataset. Evaluation is done on HPatches [1] dataset
  Architecture
  Image is divided to 64 cells in a grid 8×8, which act as bins for keypoints. Shared for descriptor and detector encoder maps the image into stacked channel–responses on this grid, thus loosing the spacial resolution, but increasing channel depth.
  The descriptors are then obtained using non–trainable bilinear interpolation and L2–normalization.
  Loss consists of two parts:

detector part, which penalizes corresponding by the chosen homography keypoints on the image and its warped version to be far. It is done with the cross–entropy loss over the cells;
descriptor part, which is the sum of hinge–like losses for inner products of descriptors of keypoints from input and warped image. Every such a loss switches between positive and negative regimes depending on how far is the warped input keypoint from the keypoint of the warped image.
Conclusion

+ Overall quality of matching using SuperPoint was shown to be better than the one of traditional methods, such that LIFT [8], SIFT [3], ORB [9].
– The descriptor of SuperPoint isn't learnable and fixed, which constrains the efficiency.
– Because the detector is trained on artificial examples, it's biased towards detecting planar geometric structures and is worse for the real world images.

- **D2–Net** [10] extends the idea of *SuperPoint* by sharing weights between the detector and descriptor and making they both learnable, therefore achieving the aim of end–to–end adaptive neural network.
Data is taken from the MegaDepth [11] dataset, which provides with ...
Descriptor part uses CNN output in 3D, which is divided to 2D–slices represented as maps–responses of channels. In order to ensure constancy of the descriptors under appearance changes, they are L2–normalized.
Detector part: conventionally output of descriptor is sparsified be means of non–maximum suppression, but in order to make it trainable soft–max function in each channel map is applied for the 9–neighborhood of the pixel. After that soft channel selection is performed by division by the maximal value for the pixel among all the channels. Both in–channel and between–channel scores are multiplied to find the channel with the maximal response. Finally, *soft detection scores* are obtained by normalisation on the image–level of these maximal products.
To take into account multi–scale detection, this procedure is applied to the image pyramid, propagating the response from lower to higher resolutions.
Loss: for simultaneous descriptor and detector training the triple margin ranking loss is used to ensure *distintiveness* of descriptors, such that correspondent descriptors are stimulated to be closer and distinct ones to be farther. The size of the "neighborhood" of the keypoint if a hyper–parameter, and therefore it defines the density of detected points.
To obtain *repeatability* the triple loss is weighted with the product of the earlier obtained soft–detection scores of compared pixels, and the weighted sum over all the putative correspondences results in the final loss.
Conclusion

+ D2–Net was shown to be consistently better than SuperPoint and the previous ones.
– It predicts properties of keypoints in rather empiric and directly following from conventional pipeline fashion, which could be not efficient when trying to implement the new detect–and–describe approach.

- **R2D2** [12] aims at tackling the problem of actual frequent uselessness of salient point calculated just as maxima of the signal, which was the main criterion in the previous methods. It proposes to estimate explicitly for each pixel its *repeatability* and *reliability* (which is almost a synonym to the used before distinctiveness).
This network produces 3 maps: **dense descriptors**, **reliability** and **repeatability**.
Data
Training is done on artificially warped images with known homographies and on one–scene images from sequence with estimated by means of the optical flow homographies. In the latter case estimation of the fundamental matrix and epipolar constraints is done, and occluded areas

are taken into account. Matching is done through using only correspondences which are in large components of the graph of connected neighbors. Images are taken from Oxford, Paris and Aachen Day–Night datasets.

Loss:

repeatability loss — in order to mitigate the negative effect of warping artifacts, borders and occlusion, repeatability heatmap is divided on squared overlapping patches, whose size is a hyper–parameter and it controls the density of detected keypoints. The loss is a weighted (another hyper–parameter) sum of two: cosine similarity, which penalizes differences in patches, and "peakiness", which ensures that the heatmaps of both compared patches are not constant and penalizes flat patches, because constant ones have perfect cosine similarity, but are meaningless;

for **reliability** loss is taken batch of ground–truth correspondences, pairwise Euclidean distances matrix is calculated and average of its rows' ranking metric — Average Precision (AP) — is calculated. After that, with the hyper–parameter such as "expected minimum AP", using its convex combination with the calculated AP, reliability is estimated as a weight of AP in this sum, such that it is pushed towards 0 when AP is low, and towards 1 when it's high.

Repeatability and reliability heatmaps are then trained simultaneously using this loss.

At test time the trained network is run on the image and its downsampled by $2^{1/4}$ versions until it's smaller than 128 pixels. K the best descriptors over all scales are kept, where the quality metric is multiplication of the repeatability and reliability values.

Conclusion

+ The quality of matching of R2D2 is at the level or better than others (including the closest and almost the same in quality D2–Net).
+ Its descriptor length is at least twice and the number of weights is at least 15 times less than the competitors have, making the model more compact.
– Needs the choice of higher number of hyperparameters.

- **UR2KiD** [13] aims at including global information into descriptors, thus making image retrieval tasks more efficient, which is crucial when working with huge image databases. This simultaneous learning of local and global descriptors in a single network pipeline can enhance both local matching and global retrieval quality.

  Data

  As training data the authors use sets of corresponding images (negative and positive correspondences). This supervision is weaker than previous ones, as no pointwise correspondences needed.

  Features

  Multi–level visual features are extracted by means of the network trained on ImageNet.

  Dimension reduction

  Inspired by D2–Net, authors suggest a little different approach: instead of looking at single descriptor channel maximum they propose to group these channels and therefore reduce dimension of data, which is beneficial for the large–scale data. In order to choose these groups in unsupervised way and prevent overfitting, they are selected by dropout on feature channels.

  Margin loss

  For any two images outer–product of their low–dimensional feature vectors result in affinity matrix, from which the average score can be calculated and adjusted according to the fact that the positive correspondence score is higher that the negative one. It results in the margin loss.

  Distillation

  However, the low–dimensional descriptor after dropout is unable to capture the whole information from the high–dimensional one. For this distillation loss is introduced, which favors the high–dimensional affinity matrix multiplied by the soft detection score (as in D2–Net) to be close to the low–dimensional affinity matrix.

  Global description

The contrastive loss with margin is then calculated on the feature maps of images, and is added to the previously considered margin losses and weighted distillation loss to obtain the final one.
Conclusion

+ UR2KiD was shown to be more robust against appearance and scale changes than the previously discussed.
+ 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.

Matching of features

Traditionally feature matching was done by means of the nearest–neighbor search, filtering and efficient data structures such as kd–tree [14] or inverted index [15]. However, they didn't use the power of neural networks which could be achieved at a low cost.
We review methods for feature matching using deep learning models:

- **Hierarchical Feature Network (HF–Net)** [16] addresses the problem of matching under computational power restrictions such as when using mobile devices. For this matching is done in the coarse–to–fine fashion.
  Hierarchical localization
  First, set of relevant images (called prior frames) is found using global descriptors and k–NN search among them.
  After that, there prior frames are clustered to "places" by similarity of 3D-structures they co–observe.
  Then for each scene 2D keypoints are matched to 3D ones from the scene and 6–DoF pose is estimated in RANSAC–like scheme.
  SuperPoint is used for local matching step, as it's more robust than conventional SIFT and more efficient to sample independently of the number of detections.
  Distillation
  However, as a self–supervised method, SuperPoint needs more data than supervised ones. Data augmentation could solve the problem, but it also can lead to the break of the global image consistency. To tackle this distillation from the teacher network is proposed. Moreover, several different teacher–networks could be used, leading to the multi–task distillation training, which could leverage the best teachers.
  As a **data** the The Aachen Day-Night [17], the RobotCar Seasons [18] and the CMU Seasons [19] datasets are used.
  Conclusion

  + HF–Net outperformed state–of–the–art 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.

- **Order–aware network (OA–Net)** [20] aims at solving the outlier rejection problem for keypoints correspondences with learning–based methods. Thus it implements the idea of using the local information about neighborhood of the keypoint. Overall, OA–Net estimates the probability of features to be inliers and after that regress essential matrix. For this it exploits PointNet–like [21] structure, which is created to find the learnable permutation–invariant clusterings of correspondences. However, its vanilla structure doesn't capture local context, such as e. g. similar motion of the neighboring pixels.

To regress the essential matrix the authors use the weighted eight–point algorithm, which is more robust to outliers in comparison to the traditional one, and which is additionally differentiable, so appropriate for the preferable end–to–end architecture.
Additionally, 3 novel operations inspired by the DiffPool [22] are introduced:

1. **Differentiable Pooling layer (DiffPool)** is PERMUTATION–INVARIANT and captures the local context by using soft learnable cluster assignments of nodes representing keypoints. Permutation–invariance means that under any permutation of the input, the resulting order of nodes, called the CANONICAL, remains the same.
2. **Differentiable Unpooling layer (DiffUnPool)** is in some sense inverse of the DiffPool. However, the inverse can't be applied directly, as going through layers the correspondences between embeddings are lost, and we need to have permutation–invariant operations. DiffUnPool upsamples the coarse representations, thus building the hierarchical structure and using localisation information.
   Another its feature that is doesn't have to take a fixed length input, which makes it more preferable at practice.
3. **Order-Aware Filtering block** is constructed in order to use that the pooled features are in the canonical order after the DiffPool layer. It includes the Spatial Correlation layer, which applies perceptrons on spatial dimension with weight–sharing along it. It's complementary to the PointNet structure, which is along the channel dimension. Therefore, Order-Aware Filtering block is constructed inserting the Spatial Correlation layer inside the PointNet.

The final structure is the common multiscale network based on DiffPool and DiffUnPool layers instead of usual ones, and the Order–Aware Filtering block in–between them.

As a **data** the YFCC100M [23] (outdoor scenes) and SUN3D [24] (indoor scenes) datasets are used. They are processed in order to avoid intersecting scenes between train and test data. Metric is the mAP.
Conclusion

+ The method shows better performance than the other contemporary ones, including extensions of the PointNet structure
+ Captures both global and local context.
– Relies heavily on descriptors, as it is an outlier–rejection network using the nearest–neighbor search.

- In **SuperGlue** [25] matching problem is represented as optimal transport problem on graphs, where nodes represent embedded descriptor+location of keypoint.
  Attention (message passsing)
  At each layer of the network the representation of the feature is changed by aggregated messages from intra– and inter–image feature connections. Each such a message is a weighted combination of the correspondent feature value with the coefficients obtained as softmax among all inner products of the query feature and the retrieved key. These queries, keys and values are also learned as linear functions of the feature representations, different at each layer of the network, but shared for all keypoints at both images. At the last layer only one vector, final matching descriptor is learned.
  Optimal assignment
  Having obtained the descriptors, similarity matrix is obtained as inner–product between all features from the first and the second images. (add about augmented assignment???)Then, after augmenting the feature space with the dustbin for poor–matching features, the optimal assignment problem is solved by Sinkhorn algorithm, the differentiable version of Hungarian algorithm.

It is run for some T iterations.

Loss

As the data ground–truth correspondences of images are used. The loss is the negative log–likelihood calculated on these ground–truth pairs with extracted from Sinkhorn algorithm probabilities.

Conclusion

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

To the best of our knowledge, there are no modern algorithms better than SuperGlue.

GPU–based approaches

Applications of matching should often be real–time, as for example in robotics, which makes it crucial to be able to compute them as fast as possible. Therefore their efficient implementations on GPUs are important, and it is crucial is to know their advantages and limitations.

- One of the specially developed libraries of machine learning algorithms for GPU is **cuML** [26]. It includes realizations of the most popular for dimensionality reduction and various models for linear and non–linear regression and classification.

- Authors of [27] describe GPU–based acceleration for visual–inertial odometry. They observe that the prominent input to the image processing latency is introduced by non–maximum suppression and consequent feature selection. The authors propose the scheme how to parallelize this operations using low, warp–level GPU computation and communication patterns. The crucial idea is to assign feature to one warp, as with processing of different features on one warp the memory accesses would be uncoalesced, leading to inefficiency. Additionally, as the image is divided between CUDA cores, the distribution of features is uniform.
  The resulting scheme achieves the average execution time several times faster than other methods. The authors evaluated non-maxima suppression, feature detection, and feature tracking timings on the EuRoC Machine Hall dataset [28].

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

- **Cascade Hashing** [29] also allows to speed up the matching process. It uses hashes obtained by dissecting the multi–dimensional feature space with random planes and decoding every bit of hash as a corresponding side of the plane where the feature lies.
  With high probability these hashes could be properly placed to several groups such that features from one group are the closest. Doing this sorting several times, the number of candidates decreases tremendously in coarse–to–fine fashion. Also the sorting using the cheap Hamming distance calculation allows to select only the closest points in the space of hashes. Only a few features at the last step need to be checked by expensively calculating their Euclidean distances to the query, and finally two closest are checked with the Lowe's ratio test.
  This approach was tested using SIFT descriptors on the Oxford [30] dataset, self–made by authors Tsinghua dataset with obtained by laser scanner ground truth data, and on the images from flickr site. Cascaded Hashing was shown to be two orders faster than the brute force matching and an order faster than the based on kd–trees *Kd–tree* and hash–based LDAHash. Moreover, the quality of matching is comparable or exceeding. Especially in comparison with the LDAHash, which has the similar idea, the Cascade Hashing is more robust to noise and is unsupervised.
  This study didn't use GPUs for the fairness of comparison with other methods, but we can

observe that as hashing is highly parallelizable, thus this approach could be used in GPU–based algorithms.

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

Practical part

As descriptors R2D2 128–bit ones were taken from two images, one is considered as query (100–10K features extracted), another is database (5K–300K descriptors).
Code is located here.

Methods used

1. **Brute-force**
   Check all $N \cdot N_{\text{db}}$ distances and find the closest to each of the query descriptors among the DB descriptors.

   For processing of 5000 descriptors with the database of $\approx 28K$ runtime of ≈ 0.5 s is achieved with GPU as in opposite to the CPU–version with ≈ 46 s runtime.

2. **Hashing**

   Further methods use [31] binned encodings $\mathbf{h}_i \in R^M$ of descriptors $\mathbf{d}_i \in R^{128}$as a binned by W distances to M random planes, whose normals $\mathbf{a}_j$ are standard–normally distributed, and shifts $\mathbf{b}_j$ are uniformly distributed in $[0, W]$:

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

   (a) **"Manhattan check in encoded space"**
   Inspired by [29] we try to shrink the number of candidates by using Manhattan distance in the space of encodings. Candidates are chosen by the closest encodings in Manhattan distance, then the closest among the candidates is chosen using Euclidean distance.

   (b) **Locally Sensitive Hashing (LSH)** [32, 31, 33]
   Obtained binned encodings are hashed using polynomial hashing mod $tableSize$. The procedure is repeated L times. The DB occurrences are structured in L linked lists (chaining) by their value in $[0, \ldots, tableSize]$.
   As a metric of quality the average Euclidean distance ("dist") in corresponding pairs is used.
   It was checked that the sent to GPU data and results of computations aren't cached (the time is indistinguishable the same with the changed query or database. The only difference is the first running of the GPU–code, as it compiles the program on GPU).

Results of the practical part

Observations for figure 1:

- 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.

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

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

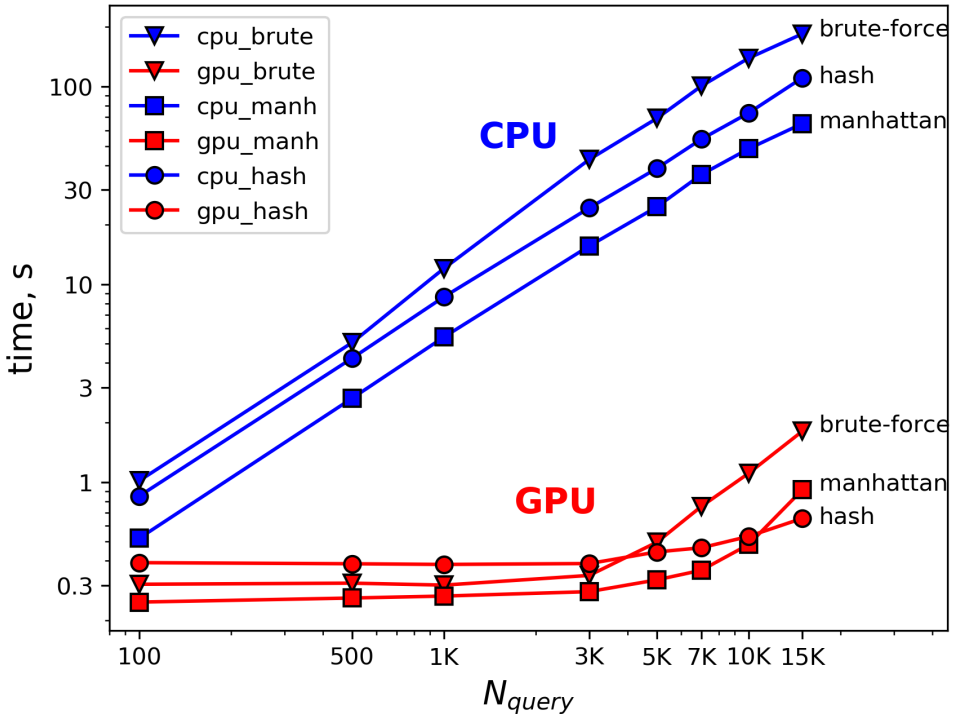

Figure 1: Execution time depending on algorithm, machine, sweeping over the **query** size.

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.

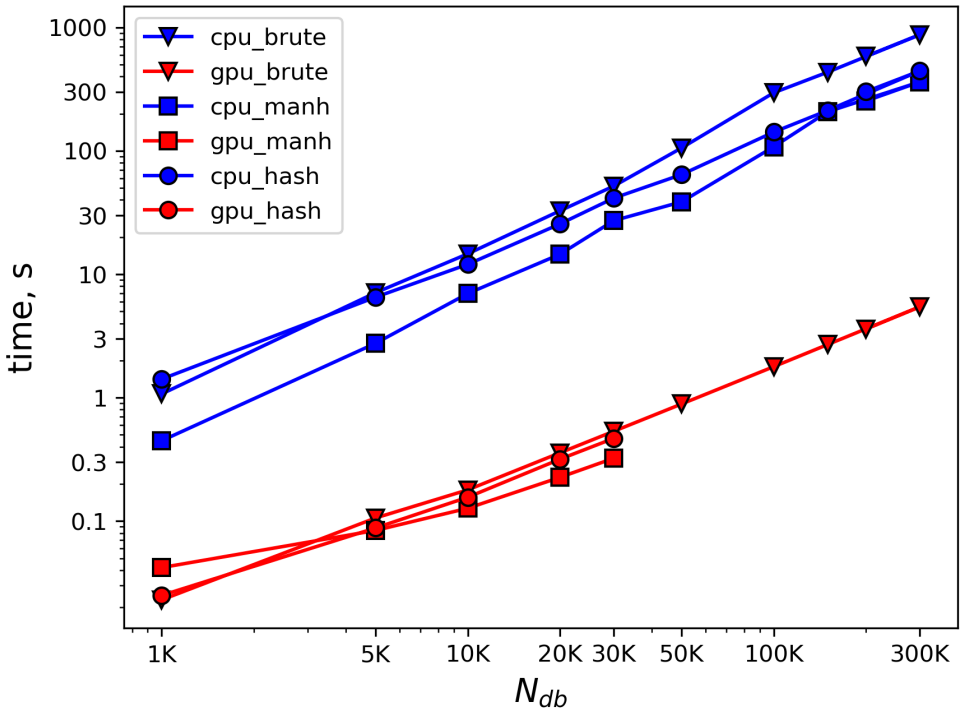

Figure 2: Execution time depending on algorithm, machine, sweeping over the **database** size. Perfect linear time dependency for both CPU and GPU, as expected.

- 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.

Figure 3: Comparison of the quality of matching for brute–force and LSH methods with some parameters. It's seen that brute–force matches perfectly the best 30 correspondences, though LSH has one mismatch (bold white line).

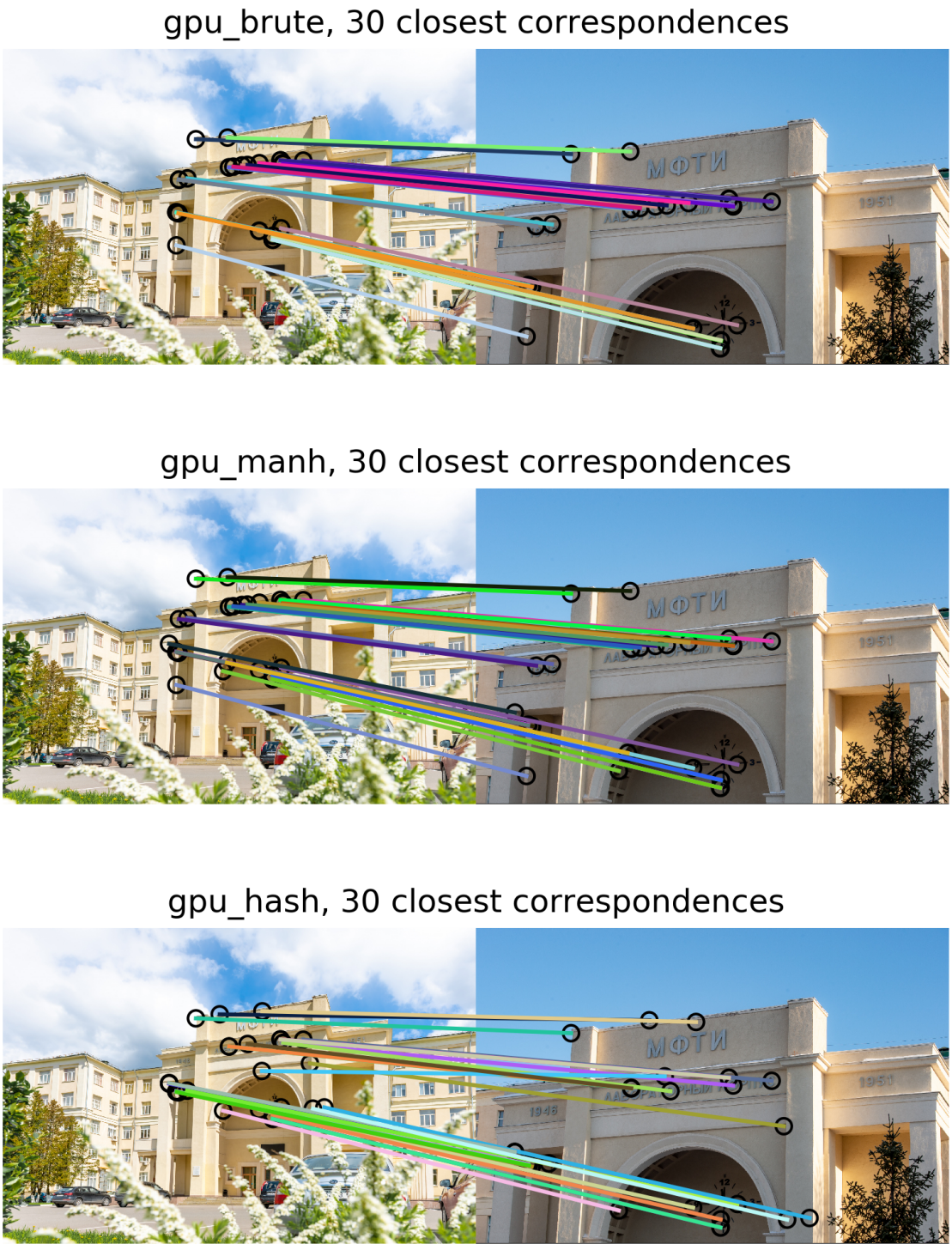

Figure 4: All three methods worked perfectly on this pair of images.

References

[1] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 04 2017.

[2] "The Visual Localization Benchmark." https://www.visuallocalization.net/.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[5] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, pp. 2548–2555, 2011.

[6] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 337–33712, 2018.

[7] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014.

[8] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 467–483, Springer International Publishing, 2016.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[10] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint detection and description of local features," *ArXiv*, vol. abs/1905.03561, 2019.

[11] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel, "R2d2: Reliable and repeatable detector and descriptor," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, eds.), pp. 12405–12415, Curran Associates, Inc., 2019.

[13] T.-Y. Yang, D.-K. Nguyen, H. Heijnen, and V. Balntas, "Ur2kid: Unifying retrieval, keypoint detection, and keypoint description without local correspondence supervision," 2020.

[14] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2d-to-3d matching," in *2011 International Conference on Computer Vision*, pp. 667–674, 2011.

[15] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," in *Robotics: Science and Systems*, 2015.

[16] P. E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 12708–12717, 2019.

[17] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image retrieval for image-based localization revisited," 2012.

[18] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Paris), 2020.

[19] A. Bansal, H. Badino, and D. Huber, "Understanding how camera configuration and environmental conditions affect appearance-based localization," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 800–807, 2014.

[20] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, H. Liao, and L. Quan, "Learning two-view correspondences and geometry using order-aware network," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 5844–5853, 2019.

[21] K. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," 11 2017.

[22] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, (Red Hook, NY, USA), p. 4805–4815, Curran Associates Inc., 2018.

[23] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research," *Commun. ACM*, vol. 59, p. 64–73, Jan. 2016.

[24] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *2013 IEEE International Conference on Computer Vision*, pp. 1625–1632, 2013.

[25] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks," 2019.

[26] "cuML - RAPIDS Machine Learning Library." https://github.com/rapidsai/cuml.

[27] B. Nagy, P. Foehn, and D. Scaramuzza, "Faster than fast: Gpu-accelerated frontend for high-speed vio," 2020.

[28] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, pp. 1157–1163, Jan. 2016.

[29] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu, "Fast and accurate image matching with cascade hashing for 3d reconstruction," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2014.

[30] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[31] J. Pan, C. Lauterbach, and D. Manocha, "Efficient nearest-neighbor computation for gpu-based motion planning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2243–2248, 2010.

[32] G. Shakhnarovich, T. Darrell, and P. Indyk, "Nearest-neighbor methods in learning and vision: Theory and practice (neural information processing)," 2005.

[33] "LSH links." https://www.mit.edu/ andoni/LSH/.