Deep Front-Ends: A quest for the state-of-the-art

Ayush Baid

Spring 2020

Contents

1 Introduction

A point-based front-end for computer vision is a system which, in a general setting, takes 2D images as input and produces an *outlier free* set of point correspondences for each pair of images. The output is also equivalent to relative geometry between camera poses. Point-based front-ends are widely used in numerous vision applications like camera calibration [1], panorama stitching [2], image retrieval [3], visual localization [4], and SLAM [5].

A good front end should have the following properties:

- Provide outlier-free (high quality) correspondences
- Provide large number of matches
- Run quickly to be usable in real time applications

We demarcate the front-end into three stages:

- **D**etection: Figure out the features (points of interest) in the input images.
- Description and **M**atching: Process features from an image pair and match features which correspond to the same 3D points. Generally, this stage is implemented in two steps: assign descriptors to input features and match them based on distance metric between the descriptors.
- **V**erification: Process the image coordinates of matches to return an outlier-free set of matches

These three stages will be referred to as **D**, **M**, and **V** respectively. There are some methods which implement the **D** and **M** stage in a single step.

There has been a lot of work in deep learning based methods for the front-end in the recent years. However, the published work often evaluate their methods independent of the other stages of the front-end, and also do not use a common set of metrics and dataset. This has led to a lot of fragmentation in the field which we want to address in our work.

We have three aims from our work:

- Perform a combinatorial search over the **DMV** to get the best possible back-end
- Provide a unified API for a front-end and the metrics ecosystem so thats its just a config change for researchers and users to use a front-end of their choice
- Standardize the metrics computed for individual stages and investigate the correlation with overall performance

2 Related Work

There have been several attempts to benchmark different components from the **DMV** pipeline. Heinly et al. [6] performed a comparison of binary **D** stage methods using two hand-crafted descriptors and evaluated the performance on feature point matching tasks, using metrics inspired by precision and recall. Schonberger et al. [7] focus primarily on the **M** stage, using a standard option for **D** and **V**. In addition to the metrics in the previous work of [6], they perform the final 3D reconstruction, a back end task, and compute metrics like re-projection error, number of points, number of images registered.

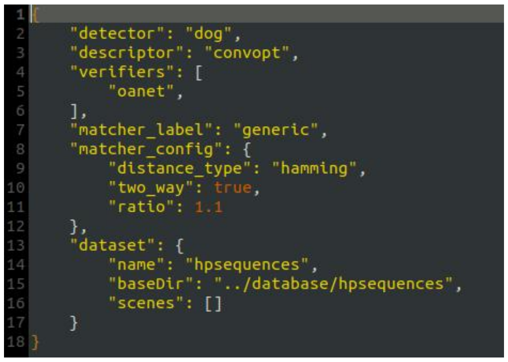
Several methods which have been proposed in recent years use their own set of evaluation metrics and dataset. We plan to perform the most comprehensive study of different combinations of front-ends, and evaluate it on all common metrics.

3 Approach

The first step of our framework is to standardize the API across each of the **DMV** stages. Given the growth in this field in the last couple of years, the code is generally not of *industry-standard* and does not have a striaghtforward API to use. It does not help that even older algorithms do not have a common interface. Our standardized function definition for the 3 stages are:

- `detect(image: 2d/3d array) -> features (2d array)`

  The features has the first two columns denoting the coordinates and an optional third column for the scale of the detected feature.
- `describe(image: 2d/3d array, features: 2d array) -> descriptors (2d array)`



```
{
    "detector": "dog",
    "descriptor": "convopt",
    "verifiers": [
        "oanet",
    ],
    "matcher_label": "generic",
    "matcher_config": {
        "distance_type": "hamming",
        "two_way": true,
        "ratio": 1.1
    },
    "dataset": {
        "name": "hpsequences",
        "baseDir": "../database/hpsequences",
        "scenes": []
    }
}
```

Figure 1: A sample JSON configuration for our software. The *detector* mentions the **D** stage, the *descriptor* and the *mactcher_config* mention the describe and match component respectively of the **M** stage, and the *verifiers* mention an array of the **V** stage methods.

`match(descriptors_1: 2d array, descriptors_2: 2d array) -> match_indices (2d array)`

The **M** stage is implemented by two functions: the first function assigns a descriptor to each input feature. The second function takes descriptors from two images as input and returns a matrix with 2 columns denoting the matches as row indices of the descriptor inputs

-
  ```
  evaluate(features_1: 2d array, features_2: 2d array) ->
          F: 3x3 matrix, final_features_1: 2d array, final_features_2: 2d array
  ```

  The function performs the geometric **V**erification to return the computed fundamental matrix along with the outlier-free set of matched features from the two images

We use this API definition and port 9 **D**, 10 **M**, 11 **DM**, and 8 **V** stage methods. This results in a total of 808 unique front-ends. This is a huge number. We need a plug-and-play configuration for faster experimentation, and for the users of our software-package to build their own front-end from one of the shipped methods or write a completely new method. Hence, we decided to use a simple JSON configuration for the front-end which specifies each of the DMV tuple, as well as the dataset to use for the benchmarking. A sample JSON configuration is presented in figure 1

As a front-end can potentially be used for different types of back-end tasks like SLAM, panorama stitching, we design a new metric which serves as a proxy for performance of back-end applications. We call this new metric *usability*. It is defined as the ratio of image pairs where the output feature points satisfy the two criteria:

- Atleast 25 points in the output. This criteria is a derived from the thumb rule of using 5 times the minimum number of points for essential matrix computation. Using a larger number of points than what is essentially required is necessary for tackling measurement errors.
- No outlier point pair in the output. We expect the front-end to verify the output and filter out the bad matches.

4 Results

We performed our experiments on two datasets: HPSequences [8] and YFCC-100M [9] (5 image pairs from each scene). The geometric relation between image pairs is homography in HPSequences and epipolar geometry in YFCC-100M. As the common use-cases in computer vision are in the 3D world with epipolar geometry, we will focus on YFCC-100M results in this report.

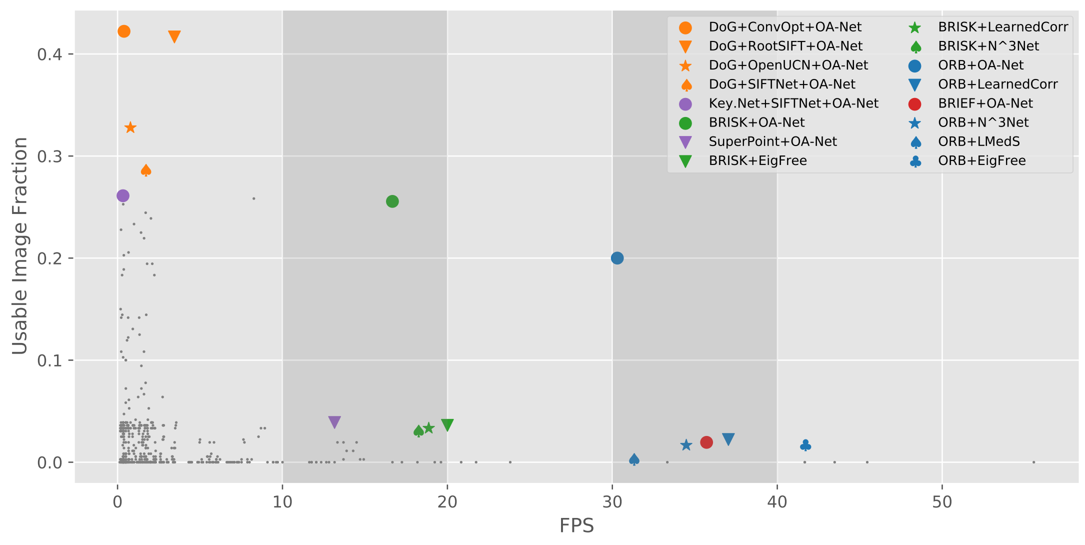


Figure 2: Competing objectives for the usability: Speed v. Usability. The plot suggests an existance of a pareto-front. We label top 5 usable methods in each bracket of 10 fps, and plot the other tuples as gray points.

We measure the usability and time of the **DMV** tuples, and present the competing performance objectives in figure 2

5 Discussion

Our experiments have some surprising results. DoG [10] + ConvOpt [11] + OA-Net [12] and DoG [10] + RootSIFT [13] + OA-Net [12] are the two best performing methods, and by a significant margin. However, a practitioner should prefer the latter method as its around 5 times faster than the former method with a marginal drop in usability. For application which have a strict real-time requirements, ORB [14] + OA-Net [12] and BRISK [15] + OA-Net [12] have a reasonable compromise between performance and speed.

We will present few pointers from our paper submitted to ECCV 2020:

- OA-Net [12] is a clear winner for **V** stage. The traditional verifier RANSAC [16] does not even feature in the top performing methods
- The top performing methods have a DoG **D** stage [10], which was introduced as a detector paired with SIFT with traditional and learnt descriptors.
- All the fast methods with some usability feature joint **DM** methods which have been here for some time: ORB [14] and BRISK [15]
- A significant portion of new deep-learning based methods are joint **DM**, and they are absent from both top usability and top performance methods

6 Future Work

There are several avenues of future work for this project:

- Release the code as an open source software. For this, we need to have 100% test coverage, and consult with a wider group to make the APIs intuitive and easy to use. We should aim to establish a new standard in front-end design, testing, and distribution.
- Integrate this front-end with an in house software like GTSAM and Shonan Rotation Averaging.
- Start a public leaderboard which keeps tracks the best front end available. It can help researchers too by providing them a way to try out combinations of their algorithm for a particular stage(s) of the front-end with other algorithms.

7 Meta-Learning

Working on this project was a interesting learning experience. Managing a large piece of software and planning different components was essential and I learnt a lot of good practises. I had peripheral knowledge of the field before this project and I gained a lot of understanding by reading the literature and I think it will be a major help for me to look for careers in computer vision and machine learning domains.

8 Acknowledgments

This work was done in collaboration with John Lambert, Alex Butenko, and Ria Verma, who are current and past students at Georgia Tech.

References

[1] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, p. 580–593, June 1997.

[2] R. Szeliski *et al.*, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2007.

[3] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2161–2168, Ieee, 2006.

[4] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative Evaluation of Binary Features," in *European Conference on Computer Vision (ECCV)*, 2012.

[7] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative Evaluation of Hand-Crafted and Learned Local Features," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *CVPR*, 2017.

[9] 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, no. 2, pp. 64–73, 2016.

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

[11] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[12] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao, “Learning two-view correspondences and geometry using order-aware network,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[13] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, IEEE, 2012.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV ’11, (Washington, DC, USA), pp. 2564–2571, IEEE Computer Society, 2011.

[15] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV ’11, (USA), pp. 2548–2555, IEEE Computer Society, 2011.

[16] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, pp. 381–395, June 1981.