Neural Feature Filtering for Faster Structure-from-Motion Localisation

Alexandros Rotsidis[1,2]
a.rotsidis@cyens.org.cy
Wang Yuxin[3]
yuxin.wang@epfl.ch
Yiorgos Chrysanthou[1]
y.chrysanthou@cyens.org.cy
Christian Richardt[2]
christian@richardt.name

[1] CYENS Centre of Excellence
Nicosia, Cyprus
[2] University of Bath
Bath, UK
[3] École polytechnique fédérale
de Lausanne, Switzerland

Abstract

Estimating a camera's pose in an offline map, i.e. camera localisation, is an important task for mobile applications such as augmented reality, self-driving cars and robotics. Many camera localisation pipelines comprise stages for feature detection, matching, outlier filtering, and solving for the camera pose. The bottleneck in localisation pipelines is typically feature matching, which becomes increasingly slower as more features are considered. This work focuses on improving feature matching speed. Specifically, we propose a neural filtering stage that reduces the number of features, drastically reducing feature matching time, with minimal loss in accuracy. This is achieved by training a scene-specific neural network to estimate how reliable (or matchable) each detected feature descriptor is. This allows us to efficiently select only the top most matchable keypoints for the remaining pose estimation pipeline. Our method is applicable to any existing structure-from-motion data. We evaluated our method on large indoor and outdoor datasets, and compare to two related methods that address the same problem. We release code of our proposed method [1].

1 Introduction

Camera localisation is the process of estimating a 6 degree-of-freedom pose matrix for an input image. This can be achieved in numerous ways; for example, a camera pose can be estimated by fetching the most similar image and using its associated pose from a database of posed images [28]. Deep learning can also regress a pose directly from an image [11]. In this paper, we focus on estimating the camera pose of a single input image against a structure-from-motion (*SfM*) map, as structure-based camera localisation tends to provide the most accurate results [29, 31]. These offline maps can be easily constructed by readily available software, such as COLMAP [32]. Descriptor matching is an important necessary stage for estimating the camera pose with respect to an offline map. In general, the process of registering an image in an SfM map follows these steps: keypoints detection, keypoint description,

[1] https://github.com/alexs7/Neural-Feature-Filtering-for-Faster-Structure-from-Motion-Localization-Code

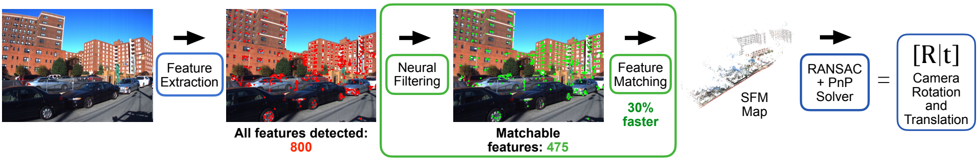

Figure 1: The proposed method augments a conventional pose estimation pipeline by adding a neural filtering stage that can filter out non-matchable features detected from a query image. The neural filtering stage can be easily plugged into existing pose estimation pipelines to efficiently select the most reliable features from all detected features. This reduces the number of features to be matched down to 30% on average (roughly 60% shown in the current example) for all datasets tested and improves feature matching times by increasing its speed while maintaining the final pose estimates.

feature matching (acquiring 2D–3D matches), outlier rejection using RANSAC [6] (or similar), and lastly solving a perspective-n-point (PnP) problem returns the camera pose. This pose estimation process typically includes two types of outlier filtering: (1) using Lowe's ratio test [4] after the feature matching stage, and (2) robustly fitting a pose matrix with a progressive or random sample consensus method [4, 6], discarding any 'outlier' 2D–3D matches that are not in the consensus set. In real-world scenarios, the thousands of keypoint descriptors detected in an image, and the large offline maps with millions of points greatly increase the computational cost of pose estimation. Each individual keypoint descriptor (n) potentially has to be checked and matched to each 3D map point (m), leading to a worst-case complexity of $O(nm)$. Increasing the number of keypoint descriptors can also lead to more outlier 2D–3D matches, which can increase RANSAC's convergence time exponentially [6].

The question we ask in this paper is: can we determine before feature matching, which descriptors are capable of being matched, thereby reducing the number of descriptors and increasing the speed of feature matching, without loss of accuracy? The contribution is a neural feature filtering method that can reduce the number of keypoint descriptors passed in the pose estimation pipeline by classifying them into matchable, e.g. static points on permanent structures such as buildings, retail shop shelves, or non-matchable, e.g. dynamic points on cars and vegetation. The network prioritize a certain number of features that can be as small as 30% of the original number of image keypoints' features, and 18% in certain cases, and can increase matching speed up to 30%, compared to existing methods. The neural network is trained on live maps, based on work by Rotsidis et al. [22]. Static keypoints, e.g. points on permanent structures such as buildings or retail shop shelves, tend to be more repeatable and thus useful in feature matching as they are more likely to be matched correctly. Our approach can be utilized in existing SfM pipeline with negligible effort, that use live map data from Rotsidis et al. [22]. The datasets we test our method on, are the CMU Extended Seasons from Sattler et al. [30], Retail shop [22], and the LaMAR dataset [25]. We compare to methods from Papadaki and Hansch [20], and Hartmann et al. [8].

2 Related Work

We start off by briefly introducing related work to keypoint detection and description. We show how keypoint detection can be accelerated before we discuss existing keypoint reduction methods.

Reducing descriptor size. One way to reduce computation and increase matching speed is via compacting descriptors with PCA-SIFT [10], which uses PCA to reduce the dimensions of the SIFT descriptor from 128 to 20, about one sixth of its original size. The number of detected descriptors remains the same for an image. The reduced descriptor size can significantly speed up individual comparisons, but for large sets of points it can still be slow [8]. Our method works differently, as we reduce the number of descriptors, not their size.

Complexity reduction methods focus on speeding up the matching process, e.g. using approximate nearest neighbors based on KD-trees [3, 17]. A vocabulary tree [19] can be used for reducing matching time [2, 26, 27]. Agarwal et al. [2] and Sattler et al. [26, 27] cluster feature descriptors into visual words and match image descriptors to the visual words. Building a vocabulary tree adds time overhead similar to training the neural networks in our proposed method. Both are a one-time preprocess only. Other approaches reduce the number of images (instead of keypoints) passed in the pose estimation pipeline using clustering [7, 14] or by parallelizing the matching process [2]. For speeding up the last stage of a pose estimation pipeline, the pose solver, alternatives to the P3P-based solver [16] have been introduced, such as *Lambda Twist* [21] and work by Nakano [18].

Reducing keypoint numbers. Knopp et al. [13] estimate a confusion score per patch in images using geographic information. This patch score is calculated by normalizing tentative feature matches to images that are more than d meters away from the query image, by the number of features detected in the patch. Descriptors detected from areas in an image with high confusion scores, such as vegetation, are avoided, hence reducing the number of keypoints to work with. Learning-based methods aim to prune correspondences [24, 34, 37] for more robust wide-baseline stereo matching (image-to-image matching), or point cloud object classification [34]. Hartmann et al. [8] proposed a hard binary random-forest classifier that is trained on structure-from-motion image pairs, and splits the keypoints from an image into 'matchable' and 'non-matchable'. Papadaki and Hansch [20] follow a similar approach, also using a random forest.

3 Methodology

In this section, we describe the design and implementation of our neural feature filtering method. We discuss the design and implementation choices we made given the available data from a SfM map, and how we use the neural network output to classify descriptors.

SfM Map Information. At the bare minimum, a SfM map contains posed images and a set of 3D points that those images observe. Each 3D point will have a list of descriptors that belong to the images' keypoints that it was observed from. A point cloud generated by SfM holds enough information that can be used to train a classifier to predict if a point is matchable or not. Most SfM software provides this information in an accessible format, such as a database. COLMAP [32], for instance, provides a list of keypoint descriptors for each image. Each keypoint is matched to a 3D point, or not if that keypoint was not triangulated. For example, a keypoint detector may detect 500 keypoints in an image, but only 100 or so may be triangulated to form 3D points when building a map, as matches need to satisfy a number of criteria to be triangulated [32]. This leads to imbalanced data, i.e. more unmatched keypoints than matched keypoints, as shown in Table 1, for each dataset. The image feature descriptors that are not matched to any 3D point are considered noise. Similarly, at query time, for localizing an image in the map, only a subset of the keypoint descriptors would

Table 1: Percentage of keypoints that can be matched to a 3D point, or not, per dataset.

Dataset	Positive (%)	Negative (%)
CMU [30]	30	70
LaMAR [25]	14	86
Retail shop [22]	60	40

match to 3D points in the map. We leverage this distilled information from the SfM map and extract all the keypoints from the SfM database. We end up with a list of SIFT descriptors [15] with their metadata, i.e. pixel location on the image, RGB value, SIFT metadata such as octave, size, orientation, response and the number of dominant orientations Papadaki and Hansch [20], as well as with a binary target value, either zero (not matched to a 3D point) or one (matched to a 3D point).

Neural Feature Filtering Network. In this section, we describe the network architecture of our Neural Filtering approach (NF), and discuss the custom cost function based on Wang et al. [36]. The inputs to the neural network are the SIFT descriptor, x and y pixel location, pixel RGB value, dominant orientations, size, response, octave, and orientation, totalling 138 features for the input vectors, one vector for each image keypoint. The neural network has six layers consisting of an input layer of 138 nodes, three more layers of 276 nodes, one layer of 138 nodes and the last output layer of one node. The activation functions in the nodes are set to ReLU [1]. The last layer activation function is a sigmoid function defined as $f(x) = \frac{1}{1+e^{-x}}$, because it produces continuous output values that are between 0 and 1, which can be interpreted as the probability that the input belongs to the positive class.

We now describe the loss function used from Wang et al. [36] to alleviate the imbalanced data problem that is shown in Table 1. Wang et al. start off by presenting a basic classifier that minimises a mean squared error (MSE). Although it is effective for balanced datasets, it is inadequate for handling imbalanced ones [36]. This is because MSE considers errors in a global sense, where it computes the loss by summing up all errors across the entire dataset, and subsequently taking the average. Wang et al. propose two cost functions called "mean false error" (MFE) together with its improved version, the "mean squared false error" (MSFE). We use the latter. The main difference between these adjusted cost functions and MSE is that MFE and MSFE calculate the average error in each class separately and then add them together. We list the MSE and MSFE functions:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}\sum_{n}\frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \quad \text{and} \tag{1}$$

$$MSFE = \frac{1}{2}((FPE+FNE)^2 + (FPE-FNE)^2), \tag{2}$$

where FPE (false positive error) and FNE (false negative error) are defined as

$$FPE = \frac{1}{N}\sum_{i=1}^{N}\sum_{n}\frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \quad \text{and} \tag{3}$$

$$FNE = \frac{1}{P}\sum_{i=1}^{P}\sum_{n}\frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2. \tag{4}$$

Table 2: The average balanced accuracy for the neural filtering proposed architecture (NF) using the binary cross entropy loss function, and the one that uses the MSFE loss function [36]. NF (MSFE) outperforms in LaMAR and Retail Shop and returns equal balanced accuracy for CMU compared to NF (BCE), using the binary cross entropy function.

Dataset	Balanced Accuracy [%]	
	NF (BCE)	NF (MSFE)
LaMAR [25]	58%	**71%**
Retail shop [22]	**63%**	**63%**
CMU [30]	65%	**67%**

The term n is the number of classes in our case $n = 2$, M is the number of all samples, N and P are the numbers of negative and positive ground-truth samples, respectively, which are already known before training. The term $d_n^{(i)}$ is the ground-truth value of the i-th sample's class, and $y_n^{(i)}$ is the predicted value of the same sample. Since $n = 2$, a simplified example is: $d^{(5)} = [0,1]$, i.e. sample number 5 belongs to the second class, and it is predicted that sample number 5, $y^{(5)} = [1,0]$ belongs to the first class. Another popular cost function used for binary classification problems is the binary cross-entropy loss (BCE) [23]. We choose the network that uses the MSFE loss function to compare to Papadaki and Hansch [20] and Hartmann et al. [8]. In Table 2, we show that the networks using the MSFE function [36] return a higher balanced accuracy than the same networks trained with the binary cross-entropy loss function, for all datasets. More details about the datasets in Section 4.

The proposed neural network is trained using Adam [12] for 1,000 epochs with a batch size of 4,096 feature vectors, SIFT and the metadata. and a learning rate of 0.0001. The CMU slices training data is, on average, 2 million rows across all slices. The LaMAR average number of training data vectors for HGE, CAB, and LIN is 16 million rows. The retail shop's training data is 3 million rows. In total, one network is trained for the retail shop, three for LaMAR, and 24 for CMU (one for each slice).

4 Experiments and Evaluation

In this section, we evaluate the proposed neural filtering method (NF), work from Hartmann et al. [8] (PM) and Papadaki and Hansch [20] (MnM). We briefly describe the datasets and metrics used, including an additional metric for this proposed work.

Datasets. We use the Retail shop dataset from Rotsidis et al. [22], and all the slices from the CMU Extended Seasons [30]. In addition to the CMU dataset and the retail shop dataset, we also used the LaMAR dataset [25]. We construct live maps using from all datasets following the same method as Rotsidis et al. [22]. Each dataset contains data captures over time of the same places, i.e. sessions. We set aside one session to use as a query session. We use the remaining sessions as additional sessions to create the live map [22], on top of the base map. All poses are in metric space.

Comparison Methods. Papadaki and Hansch [20] and Hartmann et al. [8] base their methods on a hard binary classifier model. A hard classifier is a model that makes definitive binary decisions about the class membership of a given input. We replicate their training methods and data acquisition methods. For the rest of the paper, we will use the following acronyms for our proposed method, Neural Filtering which uses the MSFE loss function (NF), and for the comparison methods, Hartmann et al. [8]'s, Predicting Matchability (PM), and Papadaki and Hansch [20], Match or No Match (MnM). For each dataset, we test one query set of images, using all three methods. We let each method decide which keypoints are matchable and which ones are not, from the detected query image keypoints. Each time we run the query images through the pose estimation pipeline, we report a number of metrics.

Metrics. We now describe the metrics we used and present the results. *Mean Rotation and Translation Errors:* There is no straightforward procedure to combine translation and rotation errors in a single combined metric. we measure the accuracy of poses similar to Sattler et al. [30]. The position error is computed as the Euclidean distance between the estimated and ground-truth camera centres, $\left\|C_{\text{est}} - C_{\text{gt}}\right\|_2$. The values are in metric. The absolute orientation error is calculated from the estimated rotation matrix $\mathbf{R}_{\text{est}}$ and the ground-truth rotation matrix $\mathbf{R}_{\text{gt}}$ exactly as in Sattler et al. [30]. *Mean Average Accuracy (mAA):* The mean errors can be misleading in cases where the results returned deviate in a significant degree. For example, a method can fail with high error numbers 49% of the time and simultaneously performs well 51% of the time with low error numbers; then the mean errors can give a false impression of perfectness. The research community now focuses on threshold-based metrics proposed in work from Jin et al. [9]. For each dataset, we use ten specific thresholds as suggested by Jin et al. [9]. The thresholds we used are, for CMU: one to 10 degrees for rotation and 0.2 meters to 5.5 meters for translation.. For Retail Shop: 0.5 degrees for rotation and one to 5 centimeters for translation. Similarly for LaMAR: one to five degrees and 0.1 meters to 0.5 meters. The mAA metric is calculated using the camera pose errors, i.e. the Euclidean distance for translation. The rotation error for mAA is calculated using the angle between the poses' quaternions p, q [9, 37]. A pose is added (or accepted) only if its rotation and translation error parts are within a threshold out of the ten thresholds. Once this process is repeated for all poses, the number of poses that fall into the thresholds is returned as a percentage value (the higher the better).

4.1 Results

We start by reporting average metrics across all datasets, and then we examine each dataset separately. Table 3 show the averaged results for CMU (24 slices), LaMAR (3 sub-datasets), and the retail shop. Since the retail shop consists of one dataset, we just report its errors, there is nothing to average. The translation is reported in centimetres (cm) for the retail shop, as the map spans four/five meters, a small shop aisle. The CMU and LaMAR span hundreds of meters, and CAB from LaMAR is also a multi-story map.

CMU Results. From Table 3, MnM returns the highest mAA for the CMU mAA = 98.99%, followed by NF, mAA = 98.54%. At the same time, our proposed method, NF, is almost *twice* as faster as the MnM, i.e. feature matching time is 66m on average for each query image (NF) but 104ms for (MnM). The method PM, although even faster than NF, the mAA is lower at 86.51%, and returns the highest average translation and rotation errors compared to NF and MnM for CMU. The task of our proposed method is to balance the reduction of keypoints (or features) while keeping low camera pose errors. NF reduces the keypoints by a further 33%,

Table 3: Mean metrics for each dataset. For CMU and LaMaR, we report the translation error in metres (m), and for the retail shop in centimetres (cm). Our neural filtering method (NF) returns that highest feature keypoints reductions, which leads to faster feature matching, and also does not deteriorate the pose errors, unlike PM which returns high reductions but also high errors.

Dataset	CMU			LaMAR			Retail shop		
Method	MnM	NF	PM	MnM	NF	PM	MnM	NF	PM
T. Er. [m/cm]	0.59	1.30	12.13	2.58	2.65	19.27	0.52	0.54	1.52
Rot. Er. [°]	0.38	0.34	10.23	2.20	2.44	41.73	0.29	0.30	0.69
Feat. Red. [%]	37.64	70.73	93.06	49.91	66.92	99.33	26.39	49.92	91.58
F.M. Time (ms)	104	66	19	2,254	1,703	45	1,020	754	142
mAA [%]	98.99	98.54	86.51	96.21	95.36	41.87	97.91	97.94	82.74

hence is faster, compared to MnM, while keeping almost the same accuracy. PM is faster than NF and MnM but fails to keep the pose errors low. Adhering to this line of reasoning, it would be unjustifiable to select the optimal technique based solely on the mean average accuracy (mAA), since the objective of this paper is not only to minimise pose estimation errors but also to reduce the feature matching time. Hence why we did not bold a single value in Table 3. In Table 4, we emphasise the mAA, feature matching time (F.M. time) and keypoint reduction percentage, for each CMU slice. Table 4, shows in bold the percentage of keypoints reduction only if the mAA for that slice is higher than the other two methods PM and MnM. The proposed method outperforms the comparison methods in 11 out of 24 CMU slices. PM outperforms NF and MnM in only one slice, i.e. CMU slice 2, being the fastest and returning a mAA increase of only 0.34%. It is worth noting that in CMU slice 19, NF filters out 82.47% of the keypoints, which is 58.56% more than the second-best performing method MnM, while returning a higher mAA.

LaMAR Results. The fastest method is PM, but also returns the lowest mAA, at 41.87%. From Table 3, NF is faster than MnM, at 1,703 milliseconds (ms) average feature matching time, 551 milliseconds faster than MnM. NF maintains low pose estimation errors and a mAA of 95.36%. MnM returns the higher mAA by only 0.85%, but it's 551ms slower. PM returns the fastest time at only 45ms average feature matching time but returns the lowest mAA, at 41.87%. In Table 5 and Table 6 we show the results for LaMAR for each method, and for each sub-dataset, LIN, CAB, HGE. The tables show that the method from Hartmann et al. [8], PM, struggles to learn the imbalanced dataset distribution of LaMAR, compared to the NF and MnM methods. Hartmann et al. [8] define their own method to get the training data and it does not account for the low number of positive examples and the high number of negative samples. The training data for PM (in LaMAR), consisted of 14% positive samples and negative 86% samples. To examine further why PM returns a lower mAA than NF and MnM we pick a random frame from LIN shown in Fig. 2. In Fig. 2, in the PM frame, the points that are matchable and true positives are drastically reduced compared to the other methods, NF and MnM. A low number of points are not desirable for pose estimation and can lead to poses with high errors [38]. Points should be spread homogeneously across the camera frame for more accurate pose estimation [5].

Table 4: In this table, we show only the keypoint reduction percentage, feature matching time (F.M time) and, mAA, for all CMU slices. We bold the keypoint reduction percentages of NF that return an equal or higher mAA than the PM and MnM.

	Keypoint Reduction[%]			F.M Time[ms]			MAA[%]		
Method	MnM	NF	PM	MnM	NF	PM	MnM	NF	PM
slice2	37.57	59.88	84.69	129	99	39	99.14	99.14	99.48
slice3	29.74	**54.83**	86.24	292	214	71	99.92	99.92	99.81
slice4	37.85	**54.95**	79.26	148	126	58	100.00	100.00	99.29
slice5	56.43	86.42	96.64	16	6	2	99.33	98.44	88.00
slice6	38.85	**60.22**	95.86	253	191	21	100.00	100.00	96.98
slice7	31.17	74.71	81.56	201	90	67	100.00	99.94	99.20
slice8	31.16	63.73	86.41	171	109	42	99.88	99.81	98.38
slice9	43.27	**75.43**	96.13	59	30	5	99.23	99.23	81.64
slice10	26.45	**61.72**	94.33	125	79	12	99.92	99.92	99.08
slice11	38.47	66.78	96.12	127	86	10	97.24	96.72	89.83
slice12	33.15	**70.25**	97.97	123	68	6	100.00	100.00	72.57
slice13	33.71	**65.78**	96.24	115	75	9	98.25	98.25	90.79
slice14	46.96	86.32	98.61	44	13	2	98.61	98.33	52.89
slice15	47.50	72.08	95.84	73	47	8	99.44	98.89	84.00
slice16	35.89	**62.90**	87.02	150	104	37	99.50	99.63	98.94
slice17	43.87	65.87	92.39	57	41	10	99.38	99.22	99.06
slice18	42.98	**63.66**	95.08	34	24	4	100.00	100.00	99.05
slice19	23.91	**82.47**	98.25	62	17	2	97.41	99.44	76.74
slice20	36.03	82.71	98.69	19	6	1	99.38	99.03	65.51
slice21	27.68	65.01	90.95	116	68	19	99.81	99.74	91.68
slice22	30.29	81.25	96.90	26	8	2	99.00	98.55	60.98
slice23	53.58	93.46	99.38	7	1	0	91.94	82.22	70.00
slice24	36.66	**75.43**	97.74	43	20	2	99.78	100.00	78.67
slice25	40.28	71.59	91.13	105	61	19	98.62	98.50	83.69

Table 5: The table shows the error metrics for the LaMAR LIN and CAB dataset.

Dataset	LaMAR LIN			LaMAR CAB		
Method	MnM	NF	PM	MnM	NF	PM
T. Er. [m]	0.03	0.03	23.03	1.99	0.81	16.23
Rot. Er. [°]	0.11	0.12	25.85	2.63	2.43	30.11
Feat. Red. [%]	56.98	65.69	99.35	51.64	73.13	98.97
F.M. Time (ms)	3,026	2,631	64	1,034	632	36
mAA [%]	98.49	98.36	56.56	95.50	94.12	47.91

Retail Shop Results. NF returns the highest mAA at 97.94% followed by MnM, which returns an mAA of 97.91%. NF is also faster than MnM by 26%, as it filters out 23.53% more keypoints than MnM. NF is the best-performing method overall for the retail shop dataset. PM follows the same trend as in the CMU and the LaMAR dataset, i.e. it discards too many true positive samples and returns the lowest mAA and highest average translation

Table 6: The table shows the error metrics for the LaMAR HGE dataset and Retail Shop. The translation error for the Retail shop is reported in centimeters, 'cm'.

Dataset	LaMAR HGE			Retail Shop		
Method	MnM	NF	PM	MnM	NF	PM
T. Er. [m/cm]	5.72	7.10	18.54	0.01	0.01	0.02
Rot. Er. [°]	3.86	4.76	69.22	0.29	0.30	0.69
Feat. Red. [%]	41.10	61.94	99.67	26.39	49.92	91.58
F.M. Time (ms)	2,701	1,844	33	1,020	754	142
mAA [%]	94.64	93.60	21.13	97.91	97.94	82.74

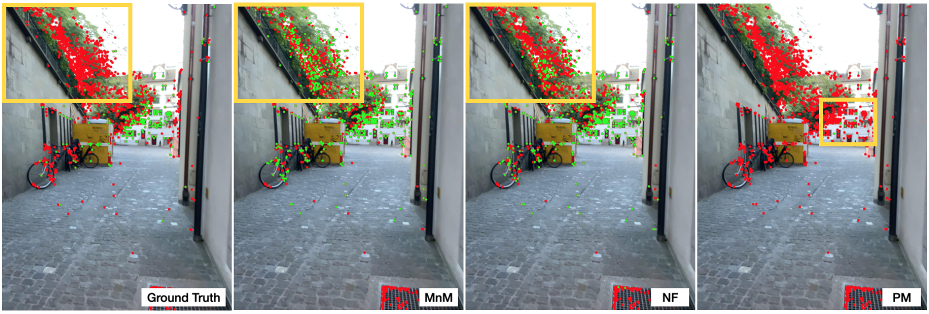

Figure 2: A random frame from LaMAR LIN, and the predictions non-matchable (red), and matchable (green). For the ground truth, the green points are the keypoints that have a 3D point matched in the live map, red if not. An area of interest is highlighted in yellow. MnM fails to detect the leaves in the top left corner as non-matchable compared to NF, which discards more points on leaves. PM fails to perform adequately and returns only a small number of points on the building.

and rotation errors compared to MnM and NF. Even though PM is the fastest from NF and MnM, its mAA is close to 15% lower than NF and MnM. In Table 5 and Table 6 we do not bold any values but we show that NF reduces keypoints by a higher margin while keeping mAA high, compared to MnM and PM.

5 Conclusion and Future Work

In this paper, we proposed and evaluated a single neural network architecture to speed up feature matching by discarding superfluous features before they are matched to an existing map. We showed that taking into consideration the imbalanced data nature of the problem, we can achieve a better balance between feature matching speed and pose estimation errors compared to existing methods. Our results show that neural networks when adapted for the imbalanced data, are a promising option for efficiently filtering out unmatchable image descriptors, which can significantly reduce downstream computation time. The experiments show that keypoints can be reduced up to 80% (in CMU slice 19) and feature matching time almost down to half while keeping the pose error minimal compared to other methods. We believe that more dynamic keypoints can be discarded while keeping the pose errors minimal if additional metadata can be added to the training data, e.g. semantic information [33, 35].

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). arXiv:1803.08375, 2018.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. In *ICCV*, pages 72–79, 2009. doi: 10.1109/ICCV.2009.5459148.

[3] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[4] Ondrej Chum and Jiri Matas. Matching with PROSAC – Progressive sample consensus. In *CVPR*, 2005.

[5] Fatih Demirtaş, Baran Gülmez, İrem Yıldırım, Uğur Murat Leloğlu, Mustafa Yaman, and Eylem Tuğçe Güneyi. Investigation of the effects of false matches and distribution of the matched keypoints on the pnp algorithm. In *Turkish National Photogrammetry and Remote Sensing Union Technical Symposium (TUFUAB)*, 2019.

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. doi: 10.1145/358669.358692.

[7] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building Rome on a cloudless day. In *ECCV*, pages 368–381, 2010.

[8] Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. In *CVPR*, pages 9–16, 2014.

[9] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 129(2):517–547, 2021.

[10] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *CVPR*, 2004.

[11] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *ICCV*, 2015.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[13] Jan Knopp, Josef Sivic, and Tomas Pajdla. Avoiding confusing features in place recognition. In *ECCV*, pages 748–761, 2010.

[14] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, pages 427–440, 2008.

[15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

[16] Xiao Xin Lu. A review of solutions for perspective-n-point problem in camera pose estimation. *Journal of Physics: Conference Series*, 1087:052009, September 2018. doi: 10.1088/1742-6596/1087/5/052009.

[17] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[18] Gaku Nakano. A simple direct solution to the perspective-three-point problem. In *BMVC*, 2019.

[19] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, volume 2, pages 2161–2168, 2006. doi: 10.1109/CVPR.2006.264.

[20] Alexandra I. Papadaki and Ronny Hansch. Match or no match: Keypoint filtering based on matching probability. In *CVPR Workshops*, pages 1014–1015, 2020.

[21] Mikael Persson and Klas Nordberg. Lambda Twist: An accurate fast robust perspective three point (P3P) solver. In *ECCV*, pages 318–332, 2018.

[22] Alexandros Rotsidis, Christof Lutteroth, Peter Hall, and Christian Richardt. ExMaps: Long-term localization in dynamic scenes using exponential decay. In *WACV*, pages 2867–2876, January 2021.

[23] Usha Ruby and Vamsidhar Yendapalli. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[24] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020.

[25] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking localization and mapping for augmented reality. In *ECCV*, 2022.

[26] Thorsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011.

[27] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, pages 752–765, 2012.

[28] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, 2012. doi: 10.5244/C.26.76.

[29] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3D models really necessary for accurate visual localization? In *CVPR*, 2017. doi: 10.1109/CVPR.2017.654.

[30] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018.

[31] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of CNN-based absolute camera pose regression. In *CVPR*, June 2019.

[32] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[33] Erik Stenborg, Carl Toft, and L. Hammarstrand. Long-term visual localization using semantically segmented images. *ICRA*, pages 6484–6490, 2018.

[34] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. ACNe: Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, pages 11286–11295, 2020.

[35] Carl Toft, Erik Stenborg, Lars Hammarstrand, Lucas Brynte, Marc Pollefeys, Torsten Sattler, and Fredrik Kahl. Semantic match consistency for long-term visual localization. In *ECCV*, 2018.

[36] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *International Joint Conference on Neural Networks*, pages 4368–4374, 2016.

[37] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, pages 2666–2674, 2018.

[38] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *ICCV*, 2013.