Data Article

Synthetic feature pairs dataset and siamese convolutional model for image matching

Houssam Halmaoui\textsuperscript{a,b,}\textsuperscript{*}, Abdelkrim Haqiq\textsuperscript{b,}\textsuperscript{*}

\textsuperscript{a} ISMAC - Higher Institute of Audiovisual and Film Professions, Rabat, Morocco
\textsuperscript{b} Hassan First University of Settat, Faculty of Sciences and Techniques, Computer, Networks, Mobility and Modeling laboratory: IR2M, Settat 26000, Morocco

A R T I C L E   I N F O

Article history:
Received 27 August 2021
Revised 31 January 2022
Accepted 10 February 2022
Available online 15 February 2022

Keywords:
Learned features
Interest points
Keypoints
Matching model
Matching pipeline
Feature descriptors
Feature patch dataset
Feature pairs

A B S T R A C T

In a previous publication [1], we created a dataset of feature patches for detection model training. In this paper, we use the same patches to create a new large synthetic dataset of feature pairs, similar and different, in order to perform, thanks to a siamese convolutional model, the description and matching of the detected features. We thus complete the entire matching pipeline. The accurate manual labeling of image features being very difficult because of their large number and the various associated parameters of position, scale and rotation, recent deep learning models use the result of handcrafted methods for training. Compared to existing datasets, ours avoids model training with false detections of the extraction of feature patches by other algorithms, or with inaccuracy errors of manual labeling. The other advantage of synthetic patches is that we can control their content (corners, edges, etc.), as well as their geometric and photometric parameters, and therefore we control the invariance of the model. The proposed datasets thus allow a new approach to train the different matching modules without using traditional methods. To our knowledge, these are the first feature datasets based on generated synthetic patches for image matching.

* Corresponding authors.

E-mail addresses: houssam.halmaoui@gmail.com (H. Halmaoui), abdelkrim.haqiq@uhp.ac.ma (A. Haqiq).

https://doi.org/10.1016/j.dib.2022.107965
2352-3409/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)
## Specifications Table

| Subject                      | Computer Vision and Pattern Recognition |
|------------------------------|----------------------------------------|
| Specific subject area        | Image matching                         |
| Type of data                 | Image                                  |
| How data were acquired       | Datasets of feature patches, feature mosaic images and feature pairs were generated using Matlab and Python programming languages on an Intel i7 computer, then, used for model training with Python on GPU (Tesla P100-PCIE-16GB). |
| Data format                  | Raw, Analyzed, Filtered, HDF5           |
| Parameters for data collection | The images were carefully selected based on the different geometric and photometric variables of rotation, angle, luminance, contrast, blur and noise. |
| Description of data collection | The patch dataset consists of about 237K images $15 \times 15$, divided into two classes of 112K features and 125K non features. From these images, a dataset of about $2M$ pairs of features is created by considering the pairs of features with the same angle and contrast as similar patches (about $1M$ pairs), otherwise they are classified as different features (about $1M$ pairs). |
| Data source location         | Institution: ISMAC - Higher Institute of Audiovisual and Film Professions, City: Rabat, Country: Morocco |
| Data accessibility           | Repository name: Mendeley Data |
| Data identification number   | 10.17632/8jx8y9yfn5.2 |
| Direct URL to data           | https://data.mendeley.com/datasets/8jx8y9yfn5/2— |
| Related research article     | H. Halmaoui, A. Haqiq, Convolutional sliding window based model and synthetic dataset for fast feature detection, in: The International Conference on Artificial Intelligence and Computer Vision, Springer, 2021, pp. 101-111. https://doi.org/10.1007/978-3-030-76346-6_10 |

## Value of the Data

- Recent image matching methods based on deep learning still use the result of traditional methods for training. This is due to the difficulty of manually and accurately labeling the position, orientation and scale of hundreds of features per image. The proposed synthetic datasets allow to avoid relying on handcrafted methods for the training of feature detection, description and matching models. Also, the use of the same patches for training the different models of the matching pipeline allows a better combination of these components.
- The datasets can be used by researchers working on one of the many applications of image matching, such as augmented reality or image registration, in order to train new models.
- The datasets are usable directly for training one or more modules of the matching pipeline. It is also possible to augment the dataset with new transformations, or to rely on the design methodology to create a new one, more specific to a certain application, for example by modifying the format and content of the images.
- The idea of using a synthetic dataset for image matching allows, by avoiding the use of traditional method results for training, and by being able to control accurately the geometric and photometric parameters of the features and generate them in very large numbers, opening up new possibilities for the training of matching models.
- The dataset simulates different geometric and photometric transformations in order to obtain matching models invariant to these transformations.
1. Data Description

We present in this article three datasets of synthetic images we have designed to train deep learning models for image matching. The first dataset, called patch dataset, is used to train convolutional and artificial feature detection models (first matching step). The second dataset, called mosaic images dataset, is used to train a convolutional sliding window model for fast feature detection. Finally, the third dataset, called feature pairs dataset, is used to train a siamese convolutional model for description and matching of the detected features (last two steps of the matching pipeline). Note that the first two datasets have already been the subject of a previous publication [1], but they will be presented here with more details about their designs, as well as some improvements that have been made. The main novelty here is the dataset of feature pairs and its use for training.

The Fig. 1 shows some random samples of the patch dataset, simulating different variations of angle, rotation, luminance, contrast, noise and blur. Fig. 3 shows an example of a mosaic

![Fig. 1. Random samples from the patch dataset with different geometric and photometric transformations. Labels 1 and 0 correspond, respectively, to features (corners) and non-features (lines and uniform regions).](image_url)
Fig. 2. Convolutional sliding window model for fast feature detection.

Fig. 3. Example of a mosaic image (top left) and the corresponding feature map before (top right) and after adding the features at the intersection (bottom right). Illustration of the new features potentially generated at the intersection of four neighboring patches (bottom left).
Fig. 4. Training of the convolutional sliding window model on mosaic dataset: MSE Loss and cosine similarity metric, according to the number of iterations, for training set and validation set.

Fig. 5. Random samples from the dataset of feature pairs. Similar pairs have a label 1 and different pairs have a label 0.
image and the corresponding feature map that allows to locate the features. The Fig. 5 shows random samples from the feature pairs dataset.

The different datasets are hosted in HDF5 format. The patch dataset is separated in two files of train set and test set in HDF5 formats. The mosaic dataset is separated into two files in HDF5 format of train and test sets. The feature pairs dataset is stored in HDF5 format in a single file.

Each dataset is accompanied by an IPYNB notebook code allowing to read and display the data as well as other information (format, type, number of samples) in order to facilitate its use.

The datasets are public and available to the community at [2].

2. Experimental Design, Materials and Methods

Feature matching consists in finding the correspondence between two images of the same scene acquired from different points of view. The applications in computer vision are numerous [3]: augmented reality, stereo, registration, camera pose, etc. Not all pixel regions of the image are reliable for matching. For example, lines and uniform regions are not locally unique in an image. Corners are more reliable features for matching [4]. The classical matching pipeline consists of 3 steps: feature detection, feature description and descriptor matching. Recent deep learning methods [5–13] use different models for each component. However, we found that these models still use feature datasets [14–18] based on the result of traditional handcrafted methods [19] for training. This is due to the difficulty to label in each image the position, orientation and scale of several hundreds of features accurately.

In fact, there are different types of datasets used to train these models. On the one hand, we have real image datasets [14–16] where features are localized by specific algorithms. On the other hand, the synthetic datasets use either real images to which synthetic geometric transformations are applied as in [17,18], or synthetic images of scenes/objects where the feature patches are labeled manually or detected by other algorithms as in [20]. The novelty in the proposed datasets, is that we have synthesized feature patches, rather than images of scenes/objects. This avoids manual labeling of features or their detection by specific algorithms, and thus allows more localization accuracy, more control over the content, and avoids the algorithm’s false detection.

Based on this observation, in order to not rely on traditional methods, we have created in a previous article [1] a new dataset of synthetic features to allow the training of feature detection models, and which will be presented here in more detail, as well as some improvements that have been made.

In this paper, in order to complete the whole matching pipeline with only deep learning models, we propose a new large dataset of feature pairs, to train a siamese convolutional model of description and matching that predicts the similarity between pairs of features.

We will start by describing the feature patch dataset used for detection [1].

2.1. Patch dataset for feature detection

In this section, we will present the image patch dataset previously used and briefly described in [1]. In the following, we will provide all the details about its design.

The idea of creating a synthetic dataset is inspired by the traditional Harris detector, still widely used today, which classifies the image into three regions: corners, lines and uniform regions. We thus generated synthetic image patches of $15 \times 15$ dimensions for each of the three shapes, in order to train a CNN and ANN models [1], that classify the image regions into feature (corners) and non-feature (lines and uniform regions), as shown in Fig. 1.

For the corner features, we simulate, with constant steps: 13 corner angles varying from $90^\circ$ to $130^\circ$ (to avoid confusing them with edges and lines), 120 rotations from 0 to $360^\circ$, 18 contrasts greater than 0.6 with corners both darker and lighter than the background, a Gaussian
noise $\mathcal{N}(0, (0.001)^2)$ (with and without), and a Gaussian blur with a kernel size $2 \times 2$ and a standard deviation $\sigma = 2/3$ (with and without). We obtain a total of features of 112320 ($13 \times 120 \times 18 \times 2 \times 2$).

For the line non-features, we simulated 90 rotations between 0 and 180° (symmetrical lines), 20 luminances and contrasts, 6 Gaussian noises with $\sigma$ between 0 and 0.002, and 3 Gaussian blurs of kernel size between 1 and 3 with $\sigma = 1$. For the half-line non-features, we simulated 180 rotations between 0 and 360°, 14 luminances and contrasts, 4 Gaussian noise with $\sigma$ between 0 and 0.002, and 3 Gaussian blurs with kernel size between 1 and 3 with $\sigma = 1$. For the uniform region non-features, we simulated 205 grey levels between 0 and 1, 34 Gaussian noises with $\sigma$ between 0 and 0.01, and 9 Gaussian blurs of kernel size between 1 and 9 with $\sigma = 3$. We obtain a total of 125370 non features.

The different values were chosen, empirically, after several tests, both to have a maximum of variabilities, visually perceptible corner and line structures, and to obtain balanced numbers of samples between the feature and non-feature classes.

The data were shuffled randomly before being stored in HDF5 format in two files of train (90%) and test set (10%).

The images are in gray level and coded on 8 bit (values between 0 and 255), it is thus necessary to normalize and convert them into float before training.

In [1], we used this dataset for training different ANN and CNN architectures, both shallow and deep, for feature detection.

For a faster detection with a convolutional sliding window model, we have created a feature mosaic dataset that we will present in the next section.

2.2. Mosaic dataset for fast feature detection

The dataset and the model that are presented in this section, are improved versions of those used in [1].

The convolutional sliding window model represented on Fig. 2 takes as input an image of dimensions $600 \times 450$ pixels and returns a feature map which allows to localize the features. It is a problem of localization by regression and not of classification as it is the case of the models using the first patch dataset. Note that the model presented here allows us to obtain a feature map that is denser than in its first version in [1]. Consequently, the dataset used is also slightly different than the one in [1].

To train this model, we have generated, from the patch dataset, mosaic images as well as the corresponding feature maps as shown in Fig. 3. The mosaic images are obtained by randomly concatenating the feature and non-feature patches in $600 \times 450$ images. Each image therefore contains 1200 patches ($40 \times 30$ patches of dimensions $15 \times 15$). We obtain a total of 198 images from the 237690 patches. The remaining patches of the last image are filled with zeros. The feature maps are obtained by assigning to the features Gaussian shapes of size $15 \times 15$ and $\sigma = 15/9$, normalized between 0 and 1. The center of the feature is therefore labeled with a value of 1 and its neighborhood with values less than 1 but not zero (because they also contain a part of the feature), this allows us to distinguish them from the non-featured patches to which we assign a value of 0. By concatenating the patches, we generate at their intersection, when one of them displays a significant contrast compared to all its neighbors, new corner features. We consider that the patch of dimension $15 \times 15$ located at the intersection of 4 patches is a feature, if one of the corners $P_i$ of the intersection (Fig. 3), satisfies the following condition:

$$\big((\text{Ctr}(M_i, M_j) > Th) \land (\text{Ctr}(M_i, M_k) > Th) \land (\text{Ctr}(M_i, M_l) > Th)\big)$$

with $\text{Ctr}(A, B)$ the contrast between two values $A$ and $B$:

$$\text{Ctr}(A, B) = \frac{\max(A, B) - \min(A, B)}{\max(A, B) + \min(A, B)}$$

$M_i$ is the median value of the corner $P_i$ of the intersection, with $i \in \{1, 2, 3, 4\}$. 
Table 1

Values of MSE and cosine similarity after training the detection model on the mosaic dataset.

|            | Loss (MSE) | Cosine Similarity |
|------------|------------|-------------------|
|            | Train      | Validation | Test     | Train      | Validation | Test     |
|            | 0.0011     | 0.0022      | 0.0023   | 0.964      | 0.9137     | 0.9395   |

j, k and l are the indices of the neighbors $P_j$, $P_k$ and $P_l$ of $P_i$, with $(i, j, k, l) \in \{1, 2, 3, 4\}$. $Th = 0.6$ is the contrast threshold used.

The mosaics and feature maps are stored in HDF5 format, in float type with values between 0 and 1, and separated in 179 images (90%) for the train set and 20 images (10%) for the test set.

We performed the model training with the following parameters: 80% train set; 10% validation set; 10% test set, epochs = 1200; Adam Optimizer; decreasing learning rate between 0.01 and 0.00001; Mean squared error or MSE loss function; Cosine similarity metric.

The training took about 7 hours on GPU (Tesla P100-PCIE-16GB).

The Fig 4 shows the learning curves and the Table 1 shows the MSE and cosine similarity values at the end of the training.

2.3. Feature pairs dataset for image matching

We present here the dataset of pairs of features, used for the training of a siamese convolutional model of feature description and matching. To build this dataset we use only the corner patches since the matching module has to find the correspondence between the detected features (corners). Note that we do not use all the corner patches because of the limits imposed by the RAM memory during the learning process as we will explain in more details later in this article. We retain only 15600 corners which allow us to generate the dataset of feature pairs. For this, we consider 13 corners as in the patch dataset, but only 10 contrasts and 30 rotations, 2 noise values, and 2 blur values.

We consider that two corners are similar if they have the same angle and the same contrast, but can have different rotations, noise and blur.

We thus build 130 subsets of similar features, each containing 120 features. Indeed, the number 120 corresponds to $30 \times 2 \times 2$ (30 rotations, 2 noises and 2 blurs). The number 130 corresponds to $13 \times 10$ (13 angles and 10 contrasts).

We note $x_j^i$ the j-th patch of the subset i of similar patches, with $i \in [1; 130]$ and $j \in [1; 120]$.

To generate the feature pairs, we arrange the subsets $X^i$ of similar patches, with $i \in [1; 130]$, as follows:

$$\{X^1, X^2, ..., X^{130}\} = \{\{x_1^1, ..., x_{120}^1\}, \{x_1^2, ..., x_{120}^2\}, ..., \{x_1^{130}, ..., x_{120}^{130}\}\}$$

From this data we generate a dataset of positive and negative feature pairs. We note $P$ the set of positive pairs (similar patches) and $N$ the set of negative pairs (different patches).

The set $P$ is constituted by all the combinations of pairs in each subset $X^i$, including the pairs constituted by the same feature:

$$P = \{(x_m^k, x_n^k) | k \in [1; 130], (m, n) \in [1; 120]\}$$

Thus, we have a number of positive samples of :

$$Card(P) = (130 \times C_{120}^2) + (130 \times 120) = 943800$$
We use the algorithm 1 for the generation of the samples of the class $P$.

**Algorithm 1** Algorithm for generating the samples of the class $P$.

1: for $k = 1; k \leq 130; k + +$ do
2: for $m = 1; m \leq 120; m + +$ do
3:  for $n = m; n \leq 120; n = ++$ do
4:  Positive sample $(x^k_m, x^k_n)$
5:  end for
6: end for
7: end for

The set $N$ of negative pairs corresponds to all the combinations of each feature of a subset $X^i$ with $i \neq j$:

$$N = \{ (x^m_p, x^n_q) | m \neq n, (m, n) \in [1; 130], (p, q) \in [1; 120] \}$$  \(6\)

So we have a number of negative samples of:

$$\text{Card}(N) = C^2_{130 \times 120} - (130 \times C^2_{120}) = 120744000$$  \(7\)

The very large number of negative pairs raises 2 problems. On the one hand, the two classes $P$ and $N$ are largely imbalanced. On the other hand, the amount of disk memory and especially RAM needed to store and process the set $N$ is about 200Gio for $15 \times 15$ patch pairs coded on 32bits.

To solve these problems we consider the set $N^*$ of negative pairs that we obtain by taking one feature out of 3 of the subset $X^i$ and one feature out of 40 of the subset $X^j$. The steps 3 and 40 were chosen in order to obtain balanced $P$ and $N^*$ classes. We used the **algorithm 2** for the generation of the samples of the class $N^*$.

**Algorithm 2** Algorithm for generating the samples of the class $N^*$.

1: for $m = 1; m \leq 130; m + +$ do
2:  for $n = m + 1; n \leq 130; n + +$ do
3:    for $p = 1; p \leq 120; p = p + 3$ do
4:      for $q = 1; m \leq 120; q = q + 40$ do
5:        Negative sample $(x^m_p, x^n_q)$
6:      end for
7:    end for
8:  end for
9: end for

We obtain $\text{Card}(N^*) = 1006200$. The two classes $P$ and $N^*$ are thus well balanced and the number of samples does not pose any problem of storage.

The total number of samples in the dataset is therefore 1.95M feature pair images. The **Fig. 5** shows some random samples.

The samples are shuffled randomly and stored in HDF5 format.

We used this dataset to train the siamese convolutional model represented on the **Fig. 6**.

The goal of the Siamese model is to differentiate patches by maximizing the Euclidean distance between the descriptors of pairs of different features:

$$\text{dist}(a, b) = ||f(a) - f(b)||_2^2$$  \(8\)

with $a$ and $b$ two features, and $f(a)$ and $f(b)$ the corresponding descriptors obtained at the output of the CNN. For that purpose, we optimize the CNNs by using a contrastive loss function that maximizes the contrast (distance) between the positive and negative classes

$$L = (1 - Y_{true})Y_{pred}^2 + Y_{true}(\max(0, \alpha - Y_{pred}))^2$$  \(9\)
with $\alpha = 1$ the margin that ensures that the distance of the negative pairs is greater than the distance of the positive pairs.

$Y_{\text{true}}$: The labels of the pairs.

$Y_{\text{pred}}$: The prediction of the model.

Note that both CNNs have the same architecture and learn the same parameters in order to compute the descriptors in the same way.

We used the following parameters for the training: 70% train set, 15% validation; 15% test set; batch size = 512; epochs = 30; decreasing learning rate between 0.1 and 0.0001; Adam Optimizer, Contrastive loss function; Accuracy metric.

The training took about 7 hours on GPU.

The Fig. 7 shows the learning curves and the table 2 the values of the contrastive loss and the accuracy at the end of the training.
Fig. 8. From top to bottom: Simulation of several transformations of $30\times$ rotation, scale 0.8, noise $\mathcal{N}(0, (0.001)^2)$, motion blur with kernel $3 \times 3$ and linear change of luminance. Left: matching of the detected features. Right: estimation of the homography from the matched features.
Table 2
Contrastive loss and accuracy after learning the matching model on the feature pairs dataset.

|          | Contrastive loss | Accuracy |
|----------|------------------|----------|
|          | Train            | Validation | Test    | Train          | Validation | Test    |
|          | 0.011            | 0.0146     | 0.0148  | 98.59%         | 98.13%      | 98.09%  |

2.4. Homography estimation

In order to test the combination of the two trained detection and matching models and how they generalize to new images, we applied different transformations to images and estimated the homography from the matched features. The matching outliers are eliminated with RANSAC. Fig. 8 shows some examples of estimation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships which have, or could be perceived to have, influenced the work reported in this article.

CRediT Author Statement

Houssam Halmaoui: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft; Abdelkrim Haqiq: Supervision, Project administration, Writing – review & editing.

References

[1] H. Halmaoui, A. Haqiq, Convolutional sliding window based model and synthetic dataset for fast feature detection, in: Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV 2021), June 28-30, 2021, Morocco. Published on the book of Advances in Intelligent Systems and Computing, 1377, Springer, 2021, pp. 101–111.

[2] H. Halmaoui, A. Haqiq, Synthetic feature datasets for image matching, mendeley data, v2, 2022, doi:10.17632/8jx8y9fyns.2.

[3] H. Halmaoui, A. Haqiq, Matchmoving previsualization based on artificial marker detection, in: Proceedings of the International Conference on Advanced Intelligent Systems and Informatics (AISI 2020), October 19, 2020, Cairo, Egypt. Published on the book of Advances in Intelligent Systems and Computing, 1261, Springer, 2020, pp. 79–89.

[4] H. Halmaoui, A. Haqiq, Feature detection and tracking for visual effects: augmented reality and video stabilization, in: Proceedings of the International Conference on Artificial Intelligence & Industrial Applications (AI2IA 2020), March 19-20, Meknes, Morocco. Published on the book of Advances in Intelligent Systems and Computing, 1193, Springer, 2020, pp. 291–311.

[5] S. Zagoruyko, N. Komodakis, Learning to compare image patches via convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4353–4361.

[6] V. Balntas, E. Riba, D. Ponsa, K. Mikolajczyk, Learning local feature descriptors with triplets and shallow convolutional neural networks, in: British Machine Vision Conference (BMVC), 1, 2016, p. 3.

[7] K.M. Yi, E. Trulls, V. Lepetit, P. Fua, Lift: learned invariant feature transform, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 467–483.

[8] Y. Tian, B. Fan, F. Wu, L2-net: deep learning of discriminative patch descriptor in euclidean space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 661–669.

[9] A. Mishchuk, D. Mishkin, F. Radenovic, J. Matas, Working hard to know your neighbor’s margins: local descriptor learning loss, in: Advances in neural information processing systems (NIPS), 2017, pp. 4826–4837.

[10] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, V. Balntas, Sosnet: Second order similarity regularization for local descriptor learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11016–11025.

[11] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, T. Sattler, D2-net: a trainable cnn for joint description and detection of local features, in: Proceedings of the ieee/cvf conference on computer vision and pattern recognition (CVPR), 2019, pp. 8092–8101.

[12] T-Y. Yang, D.-K. Nguyen, H. Heijnen, V. Balntas, Ur2kid: unifying retrieval, keypoint detection, and keypoint description without local correspondence supervision, arXiv preprint arXiv:2001.07252 (2020).
[13] P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: learning feature matching with graph neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2020, pp. 4938–4947.

[14] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Van Gool, A comparison of affine region detectors, Int. J. Comput. Vis. (IJCV) 65 (1) (2005) 43–72.

[15] M. Brown, G. Hua, S. Winder, Discriminative learning of local image descriptors, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 33 (1) (2010) 43–57.

[16] V. Balntas, K. Lenc, A. Vedaldi, K. Mikolajczyk, Hpatches: a benchmark and evaluation of handcrafted and learned local descriptors, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017, pp. 5173–5182.

[17] M. Jaher, M. Grabner, H. Bischof, Learned local descriptors for recognition and matching, Computer Vision Winter Workshop (CVWW), 2, 2008.

[18] P. Fischer, A. Dosovitskiy, T. Brox, Descriptor matching with convolutional neural networks: a comparison to sift, arXiv preprint arXiv:1405.5769 (2014).

[19] J. Ma, X. Jiang, A. Fan, J. Jiang, J. Yan, Image matching from handcrafted to deep features: A survey, Int. J. Comput. Vis. (IJCV) 129 (1) (2021) 23–79.

[20] D. DeTone, T. Malisiewicz, A. Rabinovich, Superpoint: self-supervised interest point detection and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR), 2018, pp. 224–236.