To appear in Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
The conference version will be available soon.

XFeat: Accelerated Features for Lightweight Image Matching

Guilherme Potje[1] Felipe Cadar[1,2] André Araujo[3]
Renato Martins[2,4] Erickson R. Nascimento[1,5]

[1]Universidade Federal de Minas Gerais [2]Université de Bourgogne, ICB UMR 6303 CNRS
[3]Google Research [4]Université de Lorraine, LORIA, Inria [5]Microsoft

{guipotje,cadar,erickson}@dcc.ufmg.br, renato.martins@u-bourgogne.fr, andrearaujo@google.com

Abstract

We introduce a lightweight and accurate architecture for resource-efficient visual correspondence. Our method, dubbed XFeat (Accelerated Features), revisits fundamental design choices in convolutional neural networks for detecting, extracting, and matching local features. Our new model satisfies a critical need for fast and robust algorithms suitable to resource-limited devices. In particular, accurate image matching requires sufficiently large image resolutions – for this reason, we keep the resolution as large as possible while limiting the number of channels in the network. Besides, our model is designed to offer the choice of matching at the sparse or semi-dense levels, each of which may be more suitable for different downstream applications, such as visual navigation and augmented reality. Our model is the first to offer semi-dense matching efficiently, leveraging a novel match refinement module that relies on coarse local descriptors. XFeat is versatile and hardware-independent, surpassing current deep learning-based local features in speed (up to 5x faster) with comparable or better accuracy, proven in pose estimation and visual localization. We showcase it running in real-time on an inexpensive laptop CPU without specialized hardware optimizations. Code and weights are available at `www.verlab.dcc.ufmg.br/descriptors/xfeat_cvpr24`.

1. Introduction

As a crucial step for many higher-level vision tasks, local image feature extraction remains a highly active topic of research. Despite the recent advancements, the large improvements achieved from recent image matching methods [5, 8, 40, 41] mostly come at the cost of high computational requirements and increased implementation complexity. Since image feature extraction is critical for a myriad of tasks [1, 25, 27, 29, 35, 38, 44], efficient solutions are highly desirable, especially on resource-constrained platforms such as mobile robots, augmented reality, and portable devices, where scarce computational resources are often allocated to multiple tasks simultaneously. Although specific works aim to perform hardware-level optimization for existing architectures [13], which is still hardware-specific and cumbersome in practice, few works focus on the architectural design for efficient feature extraction [46].

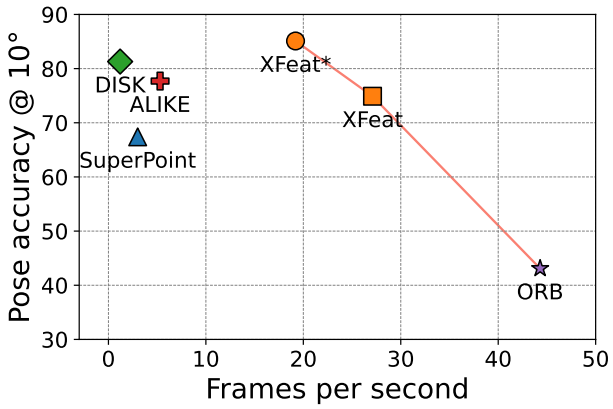

Figure 1. **In XFeat, accuracy meets efficiency.** XFeat delivers great trade-off between speed and relative pose estimation accuracy on the Megadepth-1500 dataset, as evidenced by the Pareto-frontier curve in orange. Its lightweight architecture enables real-time feature extraction on GPU-free settings and resource-constrained devices without hardware-specific optimizations. Inference speed on a budget-friendly laptop (`Intel(R) i5-1135G7 @ 2.40GHz CPU`) at VGA resolution. * denotes semi-dense extraction.

Drawing inspiration from the state-of-the-art developments in several fronts of image matching, we present **XFeat**: a novel convolutional neural network (CNN) architecture that performs keypoint detection and local feature extraction using carefully designed strategies to reduce computational footprint as much as possible, while being robust and accurate. XFeat is designed to be hardware-agnostic, ensuring broad applicability across platforms, but

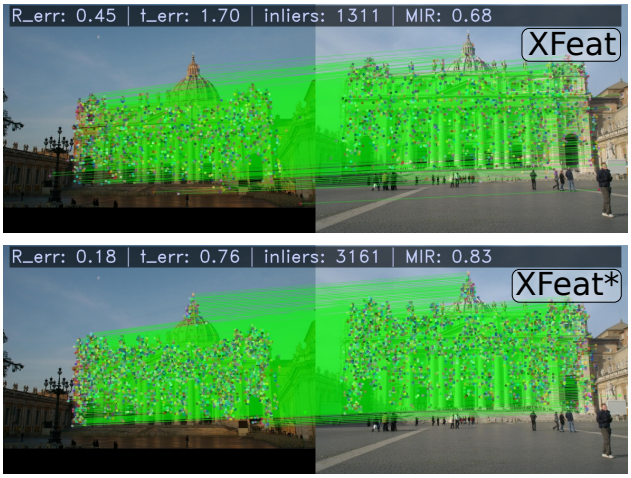

Figure 2. **Sparse (top) and semi-dense (bottom) matching.** XFeat stands out with its dual ability to perform both sparse and semi-dense matching, providing fast features for a wide range of applications from visual localization with sparse matches to pose estimation and 3D reconstruction where denser correspondences deliver additional constraints and a more complete representation.

this does not preclude the potential for optimizing XFeat on specific hardware configurations. Moreover, XFeat is suitable to perform both sparse feature matching based on keypoints and dense matching of the coarse feature map. This versatility brings the best of both worlds: keypoint-based methods are more suitable to efficient visual localization based on Structure-from-Motion (SfM) maps [35], while dense feature matching can be more effective for relative camera pose estimation in poorly textured scenes [5, 40].

Compared with current methods available for image correspondence, our method significantly improves the trade-off ratio between matching accuracy and computational efficiency as shown in Fig. 1, outperforming all lightweight deep learning local feature alternatives by up to $5\times$ in speed while being comparable to much larger models as SuperPoint [7] and DISK [42] in accuracy. To mitigate computational costs while maintaining competitive accuracy, our work brings three main contributions:

- A novel lightweight CNN architecture that can be deployed on resource-constrained platforms and downstream tasks that require high throughput or computational efficiency, without the requirement of time-consuming hardware-specific optimizations. Our method can readily replace existing lightweight handcrafted solutions [34], expensive deep models [7, 42] and lightweight deep models [46] in several downstream tasks such as visual localization and camera pose estimation;
- We design a minimalist, learnable keypoint detection branch that is fast and suitable for small extractor backbones, showing its effectiveness in visual localization, camera pose estimation, and homography registration;
- Lastly, a novel match refinement module for obtaining pixel-level offsets from coarse semi-dense matches is proposed. Our new strategy does not require high resolution features besides the local descriptors themselves as opposed to existing techniques [5, 40], greatly reducing compute and achieving high accuracy and matching density shown in Fig. 1, and Fig. 2 respectively.

2. Related Work

Image matching. Modern image matching techniques range from employing classic keypoint detection [10, 21] coupled with deep-learning based description of local patches [23, 24, 26, 30, 31], to performing joint keypoint detection and description [7, 32, 33, 42] in the same CNN backbone. More recently, middle-end approaches, known as learned matchers [14, 20, 36], and also end-to-end semi-dense [5, 40] and dense [8, 41] methods, demonstrated remarkable improvements in robustness and accuracy for matching wide-baseline image pairs, especially with the recent advances introduced by the transformer architecture [43]. However, recent methods largely emphasize image matching accuracy and robustness, thereby inflating computational demands to undesired levels, even for systems with moderate GPU resources. They require significant adaptations to work efficiently in large-scale downstream tasks such as visual localization [35], simultaneous localization & mapping [25], and structure-from-motion [38]. In contrast, in this paper we show that it is possible to drastically reduce compute utilization in both sparse keypoint extraction and pixel-level semi-dense matching, while attaining similar, or even better performances compared to more computationally expensive methods.

Efficient description & matching. Recent works highlight the growing emphasis on computational efficiency for description and matching. SuperPoint [7] proposed a self-supervised CNN for both keypoint detection and description. However, one major disadvantage of using SuperPoint is that it can still incur significant computational costs when applied to image sizes that are common for image matching. SiLK [9] reevaluates elements of learned feature extraction, proposing an effective yet simple strategy for keypoint and descriptor learning that achieves performance comparable to existing methods. The key aspect that underscores SiLK's competitiveness – its dependence on the original image size for descriptor extraction – is also its main drawback in terms of computational cost, as it substantially slows down inference. ALIKE [46] introduced a lightweight network balancing robustness and speed, with differentiable keypoint detection and a neural reprojection loss. Yet, its reliance on the original image resolution in the final feature map considerably increases memory and compute footprints. ZippyPoint [13] incorporates quantiza-

tion and binarization in a CNN. Although it achieved notable speed improvements, it requires custom compilation and specific low-level processor arithmetic operations, restricting its applicability across diverse hardware.

Works considering minimalist CNN architectures may employ both fixed handcrafted and learned filters in convolutional blocks [4]. Beyond feature extraction, recent advancements in feature matching also highlight the necessity for quick inference speeds. LightGlue [20] speeds up learnable feature matching and maintains high accuracy compared to SuperGlue [36]. Nevertheless, LightGlue's transformer-based architecture is still costly for tasks where computational efficiency is critical. In contrast to existing methods, we focus on highly-efficient and robust image matching for ubiquitous deployment: from resource-limited devices such as low-budget boards and embedded systems to smartphones and cloud applications.

3. XFeat: Accelerated Features

Local feature extraction accuracy heavily depends on input image resolution. For instance, in camera pose, visual localization, and SfM tasks, the correspondences should be fine-grained enough to allow pixel-level matches. However, feeding high-resolution images into network backbones increases computational requirements to undesired levels even for simple, small network backbones such as SuperPoint VGG-like architecture [7, 39]. In this section, we describe how to reduce significantly the computational cost using strategies to minimize the computational budget while mitigating robustness loss due to a considerably smaller CNN backbone.

3.1. Featherweight Network Backbone

Let $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ be a gray-scale image, where H is the height, W the width in pixels, and $C = 1$ denotes the number of channels. To decrease a CNN processing cost, a common approach is to start with shallow convolutions and then incrementally halve spatial dimensions (H_i, W_i) while doubling the channel count C_i in the i-th convolutional block [39]. Assuming a convolutional layer with unit stride, padding, no bias term and square kernel size $k \times k$, the cost of convolution in terms of floating point operations (F_{ops}) for the i-th layer can be expressed as:

$$F_{ops} = H_i \cdot W_i \cdot C_i \cdot C_{i+1} \cdot k^2, \tag{1}$$

Naively pruning channels C across the entire network compromises its capability of handling challenges like varying illumination and viewpoint as demonstrated in the ablation experiments (Sec. 4.4).

Efficient networks [12, 45] use depthwise separable convolutions to cut down F_{ops} by up to 9 times (with 3×3 kernel size) with fewer parameters than standard convolutions. However, in local feature extraction, where shallower networks handle larger image resolutions [7, 9, 22, 33, 46], this approach is less effective compared to their original use in low-resolution input scenarios like classification and object detection [11, 12, 39]. This leads to limited representational capacity and minor speed gains in shallow networks for local feature extraction.

In Eq. (1), the $H_i * W_i$ terms emerge as the primary computational bottleneck impacting F_{ops} in CNNs. SuperPoint [7] and ALIKE [46] reduce channel depth and layer count uniformly to alleviate the problem. We delve into the core of the issue, formulating a strategy to minimize early-layer depth and reconfigure channel distribution, significantly improving the accuracy-compute trade-off. Our proposed strategy involves reducing the channel count in initial convolution layers as much as possible due to the high spatial resolution. To counterbalance the parameter reduction, rather than adhering to the traditional VGG-like approach [39] of doubling channels, we propose tripling the channel count as the spatial resolution decreases, until a sufficient number of channels is reached (usually 128 for local feature backbones [22, 33, 42]). This strategy, marked by a triple rate increase in convolutional depth as spatial resolution halves, effectively redistributes the network's convolutional depth. It ensures minimal depth in early layers while compensating for the reduced parameter count accross the backbone. This approach not only significantly reduces the computational load in the early stages, particularly for high-resolution images, but also optimizes the network's overall capacity through more effective management of convolutional depth. We found a good trade-off between spatial accuracy and speedup gains by starting with $C = 4$ channels and concluding at $C = 128$ in the final encoder block, achieving a spatial resolution of $H/32 \times W/32$.

Our network's simplicity is anchored in blocks called basic layers, a 2D convolution with kernel sizes from 1 to 3, ReLU + BatchNorm, and a stride of 2 for resolution reduction, forming convolutional blocks, each a composite of basic layers. The backbone features six blocks, halving resolution and increasing depth in sequence: $\{4, 8, 24, 64, 64, 128\}$, plus a fusion block for multi-resolution features. More details on architecture are in the supplementary material.

3.2. Local Feature Extraction

In this section, we describe how our backbone is used to extract local features and perform dense matches.

Descriptor head. The descriptor head extracts a dense feature map $\mathbf{F} \in \mathbb{R}^{H/8 \times W/8 \times 64}$, obtained by merging multi-scale features from the encoder. By using a feature pyramid strategy [19], we inexpensively increase the receptive field of the network by applying successive convolution blocks until $1/32$ of original resolution is achieved, a strategy that

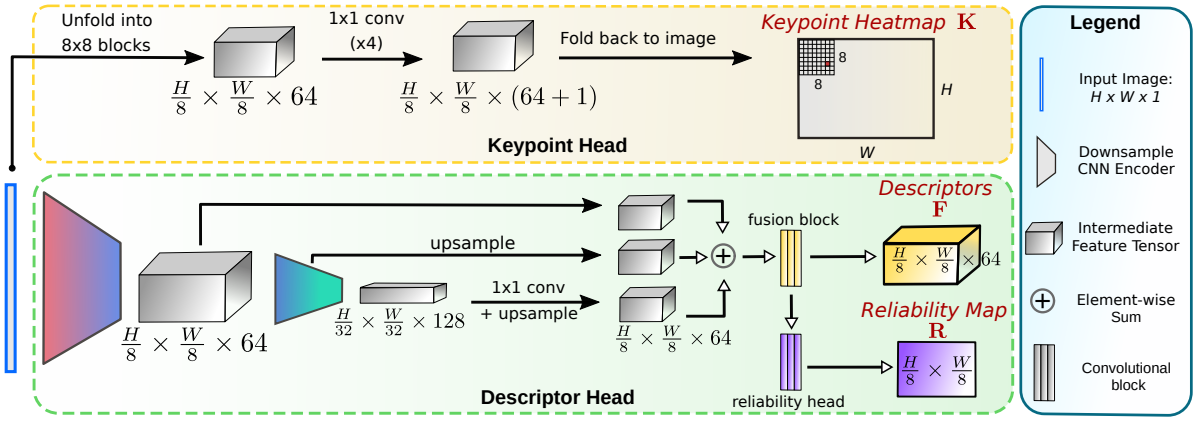

Figure 3. **Accelerated feature extraction network architecture.** XFeat extracts a keypoint heatmap $\mathbf{K}$, a compact 64-D dense descriptor map $\mathbf{F}$, and a reliability heatmap $\mathbf{R}$. It achieves unparalleled speed via early downsampling and shallow convolutions, followed by deeper convolutions in later encoders for robustness. Contrary to typical methods, it separates keypoint detection into a distinct branch, using 1×1 convolutions on an 8×8 tensor-block-transformed image for fast processing.

has demonstrated success in local feature extraction to increase robustness to viewpoint changes [22, 42, 46] and a key ingredient for small network backbones to work well in practice. We merge the intermediate representation at three different scale levels:$\{1/8, 1/16, 1/32\}$ by bilinearly upsampling and projecting all intermediate representations to $H/8 \times W/8 \times 64$ followed by element-wise summation. Finally, a convolutional fusion block composed of three basic layers is used to combine the representations into the final feature representation $\mathbf{F}$. An additional convolutional block is used to regress a reliability map $\mathbf{R} \in \mathbb{R}^{H/8 \times W/8}$, which models the unconditional probability $\mathbf{R}_{i,j}$ that a given local feature $\mathbf{F}_{i,j}$ can be matched confidently. An overview of our method is shown in Fig. 3.

Keypoint head. In general, backbones for local feature extracion rely on UNets [42], VGG [7], and ResNets [22]. The strategy used in SuperPoint [7] offers the fastest approach to extract pixel-level keypoints. It uses features in the final encoder with $1/8$ of the original image resolution, and extracts pixel-level keypoints by classifying the coordinate of the keypoint in a flattened 8×8 grid from the feature embeddings. We adopt a strategy similar to SuperPoint, but with a major difference. We introduce a novel approach that employs a dedicated parallel branch for keypoint detection focused on low-level image structures. As shown in the ablation experiments (Sec. 4.4), by jointly training a descriptor and a keypoint regressor within a single neural network backbone significantly degrades the performance of semi-dense matching for compact CNN architectures.

Our key insight lies in the efficient utilization of the low-level features through a minimalist convolutional branch. To maintain spatial resolution without sacrificing speed, we represent the input image as a 2D grid comprised of 8×8 pixels on each grid cell, and we reshape each cell into 64-dimensional features. This representation preserves spatial granularity within individual cells, while exploiting rapid 1×1 convolutions for regressing keypoint coordinates. After four convolutional layers, we obtain a keypoint embedding $\mathbf{K} \in \mathbb{R}^{H/8 \times W/8 \times (64+1)}$ encoding the logits of keypoint distribution inside a cell $\mathbf{k}_{i,j} \in \mathbf{K}$, and classify the keypoint as one of the 64 possible positions inside $\mathbf{k}_{i,j} \in \mathbb{R}^{65}$ plus a dustbin to consider the case where no keypoint is found [7]. During inference, the dustbin is discarded and the heatmap is re-interpreted as an 8×8 cell. Fig. 3 depicts the entire process of the Keypoint Head.

Dense matching. Recent research [5, 40] has demonstrated the benefits of dense image region matching, improving coverage and robustness. Our work proposes a lightweight module for dense feature matching, differing from other detector-free methods in two ways. Firstly, we can control memory and compute footprint by selecting top-K image regions according to their reliability score $\mathbf{R}_{i,j}$ and caching them for future matching. Secondly, we propose a simple and lightweight Multi-Layer Perceptron (MLP) to perform coarse-to-fine matching without high-resolution feature maps [9, 40], enabling us to perform semi-dense matching in resource-constrained settings.

Given the dense local feature map $\mathbf{F}$, which is at $1/8$ of input spatial resolution, or a subset $\mathbf{F}_s \in \mathbf{F}$, we propose a simple refinement strategy to recover pixel-level offsets. Let $\mathbf{f}_a \in \mathbf{F_1}$ and $\mathbf{f}_b \in \mathbf{F_2}$ be two matching features obtained by traditional nearest neighbor matching from an image pair $(\mathbf{I_1}, \mathbf{I_2})$. We predict offsets $\mathbf{o} = \text{MLP}(\text{concat}(\mathbf{f}_a, \mathbf{f}_b))$, classifying the offset (x, y) that leads to the correct pixel-level match at original image resolution:

$$(x, y) = \underset{\substack{i \in \{1,\dots,8\} \\ j \in \{1,\dots,8\}}}{\arg\max}\ \mathbf{o}(i, j), \tag{2}$$

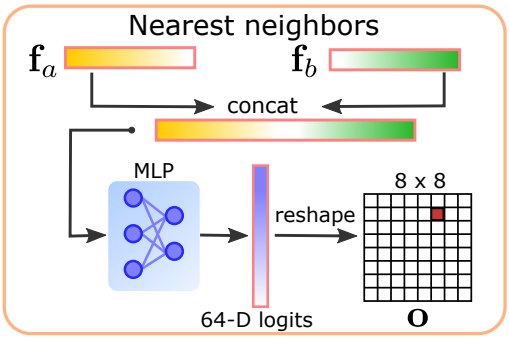

Figure 4. **Match refinement module for dense matching setting.** This module learns to predict pixel-level offsets by only considering as input pairs of nearest neighbors from the original coarse-level features at $1/8$ of original spatial resolution, significantly saving memory and compute.

where $\mathbf{o} \in \mathbb{R}^{8\times8}$ has the logits of a probability distribution over the possible offsets.

The match refinement module is trained in an end-to-end manner alongside the backbone network, ensuring that the intermediate feature representation retains fine-grained spatial details within a compact embedding space. The offset prediction is conditioned on the coarsely matched feature pair $(\mathbf{f}_a, \mathbf{f}_b)$, reducing the search space. Fig. 4 illustrates the lightweight match refinement module.

3.3. Network Training

We train XFeat in a supervised manner with pixel-level ground truth correspondences. We assume image pairs $(\mathbf{I}_1, \mathbf{I}_2)$ with N matching pixels $\mathbf{M}_{I_1\leftrightarrow I_2} \in \mathbb{R}^{N\times4}$, where the first two columns of $\mathbf{M}_{I_1\leftrightarrow I_2}$ encode the (x, y) coordinates of the points in $\mathbf{I}_1$, and the last two columns for $\mathbf{I}_2$.

Learning local descriptors. To supervise the local feature embeddings $\mathbf{F}$, we employ the negative log-likelihood (NLL) loss. Descriptor sets $\mathbf{F}_1$ and $\mathbf{F}_2$ are sampled from the dense maps $\mathbf{F}_{(\cdot,\cdot)}$, and each is represented in $\mathbb{R}^{N\times64}$, comprising N 64-dimensional descriptors. The i-th rows $\mathbf{F}_1(i,\cdot)$ and $\mathbf{F}_2(i,\cdot)$ correspond to two descriptors of the same point from $\mathbf{I}_1$ and $\mathbf{I}_2$ respectively. Then, the similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ is obtained by: $\mathbf{S} = \mathbf{F}_1\mathbf{F}_2^{\mathsf{T}}$. Given the symmetry of matching, we take both matching directions [40], resulting in the dual-softmax loss $\mathcal{L}_{ds}$, where the similarity measure of corresponding features lie in the main diagonal $\mathbf{S}_{ii}$ of $\mathbf{S}$ and softmax_r is performed row-wise:

$$\mathcal{L}_{ds} = -\sum_i \log(\text{softmax}_r(\mathbf{S})_{ii}) - \sum_i \log(\text{softmax}_r(\mathbf{S}^{\mathsf{T}})_{ii}). \quad (3)$$

Learning reliability. We supervise the reliability map during training by interpreting the dual-softmax probability as a confidence measure, denoted as $\bar{\mathbf{R}} \in \mathbb{R}^N$. $\bar{\mathbf{R}}_1$ and $\bar{\mathbf{R}}_2$ are obtained by matching $\mathbf{F}_1$ and $\mathbf{F}_2$ with the dual-softmax strategy: $\bar{\mathbf{R}}_1 = \max_r(\text{softmax}_r(\mathbf{S}))$, and $\bar{\mathbf{R}}_2 = \max_r(\text{softmax}_r(\mathbf{S}^{\mathsf{T}}))$, similarly to Eq. (3). As the training progresses, intuitively, distinct features will have high confidence matching probability. Thus, we supervise the reliability map directly with the L1 loss given the dual softmax scores $\bar{\mathbf{R}}_1$ and $\bar{\mathbf{R}}_2$:

$$\mathcal{L}_{rel} = |\sigma(\mathbf{R}_1) - \bar{\mathbf{R}}_1 \odot \bar{\mathbf{R}}_2| + |\sigma(\mathbf{R}_2) - \bar{\mathbf{R}}_1 \odot \bar{\mathbf{R}}_2|, \quad (4)$$

where σ is the sigmoid activation function and $\odot$ the Hadamard product. Note that for the reliability loss $\mathcal{L}_{rel}$, we only backpropagate the gradients through $\mathbf{R}$.

Learning pixel offsets. The match refinement module is supervised with pixel-level offsets obtained from the ground-truth correspondences $\mathbf{M}_{I_1\leftrightarrow I_2}$ at the original input image resolution. We also employ the NLL loss over the logits $\mathbf{o}$ described in Sec. 3.2. During training, corresponding descriptors $\mathbf{F}_1(i,\cdot)$ and $\mathbf{F}_2(i,\cdot)$, together with their ground-truth offset $(\bar{x}, \bar{y})$ are obtained using $\mathbf{M}_{I_1\leftrightarrow I_2}(i,\cdot)$, and the fine matching loss $\mathcal{L}_{fine}$ becomes:

$$\mathcal{L}_{fine} = -\sum_i \log(\text{softmax}\,(\mathbf{o}_i))_{\bar{y}_i,\bar{x}_i}. \quad (5)$$

Learning keypoints. Our keypoint detection branch is minimalist by design. Whilst it is possible to supervise the keypoint head with existing keypoint losses [33, 42, 46], we chose to employ knowledge distillation from a larger teacher network to facilitate its learning. We opted for ALIKE [46] keypoints obtained from its tiny backbone to supervise our model. This choice is strategic, as the smaller backbone tends to concentrate on lower-level image features like corners, lines, and blobs, aligning well with our designed detector branch, given its limited receptive field size of 8×8 pixels. Given the keypoint raw logit map $\mathbf{K} \in \mathbb{R}^{H/8\times W/8\times(64+1)}$, we map keypoint coordinates from the teacher network (t_x, t_y) inside each cell $\mathbf{k}_{i,j} \in \mathbb{R}^{65}$ to linear index $t_{idx} = (t_x + t_y * 8), \quad t_{idx} \in \{0, 1, ..., 63\}$. To supervise the dustbin, when no keypoint is detected inside a cell $\mathbf{k}_{i,j}$, we set $t_{idx} = 64$. During training, we set an upper limit of samples for the no keypoint case to avoid class imbalance. Finally, the NLL loss is employed to compute the keypoint loss $\mathcal{L}_{kp}$:

$$\mathcal{L}_{kp} = -\sum_k \log(\text{softmax}\,(\mathbf{k}_{i,j}))_{t_{idx}}. \quad (6)$$

The final loss $\mathcal{L}$ is then a linear combination of all losses:

$$\mathcal{L} = \alpha\mathcal{L}_{ds} + \beta\mathcal{L}_{rel} + \gamma\mathcal{L}_{fine} + \delta\mathcal{L}_{kp}, \quad (7)$$

where $\{\alpha, \beta, \gamma, \delta\}$ are hyperparameters to adjust the magnitude of the different losses.

4. Experiments

We evaluate XFeat on relative camera pose estimation, visual localization, and homography estimation. We also present ablations to justify our design decisions, and a comprehensive runtime analysis in a GPU-free setting.

Training. XFeat was implemented on PyTorch [28] and trained on a blend of Megadepth [17] and synthetically warped COCO [18] images, using a 6:4 ratio, with images resized to $(W = 800, H = 600)$. Hybrid training was found to enhance generalization in our experiments (Sec. 4.4), aligning with recent findings [20]. The training involved batches of 10 image pairs using the Adam optimizer [15], leading to convergence within 36 hours on an NVIDIA RTX 4090 GPU. Further details on computational resource utilization and hyperparameter specifics are provided in the supplementary material.

XFeat inference. We considered two settings: Sparse (XFeat) and semi-dense matching (XFeat*), both utilizing the same pretrained backbone. In XFeat, we extracted up to 4,096 keypoints from the keypoint heatmap $\mathbf{K}$, using their scores derived from the keypoint and reliability confidences: $score = \mathbf{K}_{i,j} \cdot \mathbf{R}_{i,j}$. Local features were then bicubically interpolated from $\mathbf{F}$ at these keypoint locations and matched with Mutual Nearest Neighbor (MNN) search. For XFeat*, we enhanced features by processing images at 2 different scales (0.65 and 1.3, resizing the image internally after receiving the input), retaining the top 10,000 features according to their reliability. We used MNN search and offset refinement to match the features, retaining only those with offset prediction confidence above 0.2.

Baselines. Among the selected baselines, DISK [42] sets a high benchmark in accuracy at the cost of increased computational demand. This is followed by SiLK [9], SuperPoint [7], ZippyPoint [7], and ALIKE [46]. For SiLK and ALIKE, we opted for their smallest available backbones – *ALIKE-Tiny* and *VGGnp-μ* – aligning with our focus on models emphasizing compute efficiency. Finally, ORB [46] represents the upper limit in terms of speed. Thus, we evaluate XFeat against the current state-of-the-art through a diverse set of baselines covering the spectrum of computational expense and accuracy. We use the top $4,096$ detected keypoints for all baselines, except for those marked with *, where the top $10,000$ keypoints are used. For matching, MNN search is employed. ZippyPoint model was used in its form as provided by the authors without hardware-specific compilation, due to the lack of clear instructions.

4.1. Relative pose estimation

Setup. Megadepth [17] and ScanNet [6] test sets are used as in previous works [20, 40], providing camera poses on scenes that do not overlap with our training set. The scenes contain significant viewpoint and illumination changes simultaneously and present repetitive structures, posing a significant challenge. LO-RANSAC [16] is used to estimate the essential matrix. We search for the optimal threshold for each method, and resize the images such that the maximum dimension becomes 1,200 pixels for Megadepth and use the default (VGA) resolution for ScanNet.

Metrics. We use the area under the curve (AUC) at thresholds of $\{5°, 10°, 20°\}$ [20, 40]. Additionally, we report the Acc@10°, which is the proportion of poses where the maximum angular error is below 10 degrees, the mean inlier ratio (MIR), which is the ratio of matching points that comply with the estimated model after RANSAC, and the number of inlier points (#inliers). Finally, we measure the frames per second (FPS) of each method on a budget-friendly laptop without GPU and an Intel(R) i5-1135G7 @ 2.40GHz CPU. We also indicate whether the descriptor is floating-point (denoted by **f**) or binary-based (denoted by **b**) and report the descriptor dimensionality (dim).

Results. Tab. 1 shows the metrics on the relative camera pose estimation task on Megadepth-1500. Our method is much faster (5×) than the fastest available learning-based solution (ALIKE) and achieves competitive results in the sparse setting on several metrics. Moreover, it can deliver state-of-the-art results for the dense matching configuration using $10,000$ descriptors on AUC@20°, Acc@10° and MIR in a fair comparison with DISK*, a much heavier model, considering the same number of descriptors. Fig. 5 shows examples where XFeat stands out over existing solutions. Our method also allows more efficient matching with low-dimensional descriptors (64-f) compared to DISK and SuperPoint. Detailed timing analysis is provided in the supplementary material alongside additional quantitative comparison with recent popular learned matchers [20, 40, 47]. It is worth mentioning that we obtain state-of-the-art results in more loose thresholds due to the requirement of interpolating the descriptors and predicting offsets at coarser resolution. Tab. 2 shows AUC values for the most competitive methods in ScanNet-1500 indoor imagery. Notice that none of the methods were retrained. DISK and ALIKE show signs of bias towards landmark datasets, while our approach demonstrates superior generalization. A more detailed discussion and qualitative results for ScanNet and Megadepth are available in the supplementary material.

4.2. Homography estimation

Setup. We used the widely adopted HPatches [2] dataset containing sequences of images from planar scenes with moderate to strong viewpoint and illumination changes. Similarly to relative pose estimation, we used MAGSAC++ [3] to robustly estimate the homography transformation given the correspondences of each method.

Metrics. We followed ALIKE [46] protocol and estimated Mean Homography Accuracy (MHA). We used pre-

Table 1. **Megadepth-1500 relative camera pose estimation.** Our method achieves superior performance compared to other lightweight methods, while also outperforming SuperPoint at 9× speedup, and with comparable results to DISK at 16× speedup. * denotes 10k keypoints. FPS is the average of 30 frames ± standard deviation computed in VGA resolution. Best in bold, second best underlined, separated by method class (standard/fast). + indicates code used as provided by authors without hardware optimization.

	Method	AUC@5°	AUC@10°	AUC@20°	Acc@10°	MIR	#inliers	dim	FPS
Standard	SiLK [9]	14.7	21.5	29.3	31.9	0.17	235	**32-f**	2.8 ± 0.08
	SiLK* [9]	16.2	23.2	31.8	34.7	0.14	478	**32-f**	2.9 ± 0.12
	SuperPoint [7]	37.3	50.1	61.5	67.4	0.35	495	256-f	$\mathbf{3.0 \pm 0.07}$
	DISK [42]	53.8	65.9	75.0	**81.3**	**0.72**	1231	128-f	1.2 ± 0.01
	DISK* [42]	**55.2**	**66.8**	**75.3**	**81.3**	0.71	**1997**	128-f	1.2 ± 0.01
Fast	ORB [34]	17.9	27.6	39.0	43.1	0.25	288	**256-b**	$\mathbf{44.3 \pm 1.18}$
	ZippyPoint [13]	23.6	34.9	46.3	51.8	0.23	192	**256-b**	$^{+}1.8 \pm 0.06$
	ALIKE [46]	49.4	61.8	71.4	77.7	0.47	333	64-f	5.3 ± 0.33
	XFeat	42.6	56.4	67.7	74.9	0.55	892	64-f	27.1 ± 0.33
	XFeat*	**50.2**	**65.4**	**77.1**	**85.1**	**0.74**	**1885**	64-f	19.2 ± 1.12

Figure 5. **Qualitative results on Megadepth-1500.** XFeat* and XFeat demonstrate exceptional robustness against variations in viewpoint and illumination. This is especially evident in challenging scenarios where heavy methods like DISK* breaks and XFeat* provide accurate relative pose 16× times faster in semi-dense settings with a comparable number of local features.

Table 2. **ScanNet-1500 relative pose estimation.** XFeat and XFeat* exhibit better generalization to indoor scenes.

AUC	Super-Point	DISK (4k/10k)	ORB	ALIKE	XFeat/ XFeat*
@5°	12.5	9.6 / 11.3	9.0	8.0	16.7 / **18.4**
@10°	24.4	19.3 / 22.3	18.5	16.4	32.6 / **34.7**
@20°	36.7	30.4 / 33.9	29.9	25.9	47.8 / **50.3**

defined thresholds of $\{3, 5, 7\}$ pixels. The accuracy was computed considering the average corner error in pixels by warping the four reference image corners to target images using the ground-truth homography and estimated one.

Results. Tab. 3 shows that our method is on par with the most accurate descriptors, reinforcing the robustness of our proposed keypoint and descriptor heads. In contrast, the performance of other lightweight solutions as ORB and SiLK are heavily compromised on the illumination and viewpoint splits, due to their limited capacity in handling agressive viewpoint and illumination changes present in the hardest image pairs. Our method also stands out for less strict thresholds, as discussed in Sec. 4.1 – Results.

4.3. Visual localization

Setup. The hierarchical localization pipeline HLoc [35] is used to localize images of day and night scenes from the Aachen dataset [37]. Given the provided keypoint correspondences, HLoc triangulates an SfM map using the available ground-truth camera poses. A separate set of query images are then localized within the 3D map using the keypoint matches. For a fair comparison, we resize the images such that maximum dimension is held at $1,024$ pixels, and extract the top $4,096$ keypoints for all approaches.

Metrics. We use the standard metric provided by HLoc, which is the accuracy of correctly estimated camera poses

Table 3. **Homography estimation on HPatches.** All methods perform well due to RANSAC except ORB and SiLK which break on several illumination sequences. XFeat provides high quality homography estimation with a fraction of compute. Best in bold, second best underlined, separated by standard and fast methods.

Method	Illumination MHA @3	Illumination MHA @5	Illumination MHA @7	Viewpoint MHA @3	Viewpoint MHA @5	Viewpoint MHA @7
SiLK	78.5	82.3	83.8	48.6	59.6	62.5
SuperPoint	**94.6**	98.5	98.8	**71.1**	**79.6**	**83.9**
DISK	**94.6**	**98.8**	**99.6**	66.4	77.5	81.8
ORB	74.6	84.6	85.4	63.2	71.4	78.6
ZippyPoint	94.2	96.9	98.5	66.1	76.8	80.7
ALIKE	94.6	**98.5**	**99.6**	68.2	77.5	81.4
XFeat	**95.0**	98.1	98.8	**68.6**	**81.1**	**86.1**

Table 4. **Visual localization on Aachen day-night.** XFeat enables fast and accurate localization, especially on the more challenging case of matching day-to-night images, being on-par with the state-of-the-art on thresholds above $0.5m, 5^\circ$. Best in bold, second best underlined, separated by standard and fast methods.

Method	Day $0.25m$ 2°	Day $0.5m$ 5°	Day $5m$ 10°	Night $0.25m$ 2°	Night $0.5m$ 5°	Night $5m$ 10°
SuperPoint	**87.4**	93.2	97.0	77.6	85.7	95.9
DISK	86.9	**95.1**	**97.8**	**83.7**	**89.8**	**99.0**
ORB	66.9	76.1	81.7	10.2	12.2	19.4
ZippyPoint	80.7	88.6	93.7	61.2	70.4	79.6
ALIKE	**85.7**	**92.4**	**96.7**	**81.6**	88.8	**99.0**
XFeat	84.7	91.5	96.5	77.6	**89.8**	98.0

within thresholds of position errors $\{0.25m, 0.5m, 5m\}$ and rotation errors $\{2^\circ, 5^\circ, 10^\circ\}$ respectively.

Results. Table 4 presents the results of the visual localization experiment. Our method demonstrates similar performance to leading approaches as SuperPoint and DISK, while achieving a significant speed advantage, being at least 9 times faster and with a more compact descriptor. These findings challenge the prevailing trend in the literature to employ large and more intricate models for downstream tasks. Contrarily, they underscore the efficacy of simpler models that not only match accuracy but also offer the benefits of efficient operation on resource-constrained systems.

4.4. Ablation

We ablate our architecture in several configurations, which are listed in Tab. 5. We evaluate whether training with additional synthetic warps (i) may help the model to become more robust to challenging image pairs. Training on both real and synthetic warps is beneficial, especially for the dense matching setting. Second, we evaluate if we can further reduce channel count in the network (ii). We halve the channels of the last three convolutional blocks to 32 instead of 64, but performance significantly degrades for both sparse and dense settings. We also demonstrate the rationale behind devising a parallel branch for keypoint detection. Without the proposed keypoint head (iii), an additional convolutional block is used on top of the output descriptor embeddings akin to SuperPoint. As shown in Tab. 5, XFeat* experiences degradation in performance when trained under this specific setup, since the limited network size constrains the capacity of intermediate embeddings, rendering them less effective for semi-dense matching in non-repeatable regions, adversely affecting the match refinement task. Thus, we opted to design a parallel branch which offers great trade-off between sparse and dense matching as shown in Tab. 5 – Default and (iii). Lastly, we evaluate the benefits of our proposed match refinement module. For XFeat*, the match refinement step is critical for enhancing accuracy. In our benchmarks, this module incurs only an additional 11% inference cost compared to MNN matching for an average of 10,000 descriptors. Please check the supplementary material for a thorough timing analysis.

Table 5. **Ablation on Megadepth-1500.** We ablate the architecture and training strategies for relative camera pose estimation.

Strategy	AUC@5° XFeat	AUC@5° XFeat*
Default	42.6	50.2
(i) No synthetic data	41.5	33.9
(ii) Smaller model	37.4	40.7
(iii) Joint keypoint extraction	42.9	39.7
(iv) No match refinement	-	38.6

5. Conclusion

This work introduced XFeat, a lightweight CNN architecture for accelerated feature extraction, applicable to both sparse and semi-dense image matching. With experiments on three different tasks and ablation analyses, we showed that it is possible to achieve fast and accurate image matching without resorting to advanced low-level hardware optimizations. This stands in contrast to the prevalent trend of deploying increasingly large and convoluted models. We believe XFeat paves the way for next-generation applications in augmented reality and mobile robotics, where efficient and general data-driven solutions remain crucial for real-world deployment, particularly in mobile applications.

Acknowledgements. This work was supported by CAPES, CNPq, FAPEMIG, Google and ANR (ANR-23-CE23-0003-01), to whom we are grateful.

References

[1] D. Aiger, A. Araujo, and S. Lynen. Yes, we CANN: Constrained Approximate Nearest Neighbors for local feature-based visual localization. In *ICCV*, 2023. 1

[2] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, pages 5173–5182, 2017. 6

[3] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *CVPR*, pages 1304–1312, 2020. 6

[4] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key. net: Keypoint detection by handcrafted and learned cnn filters. In *ICCV*, pages 5836–5844, 2019. 3

[5] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. In *ECCV*, pages 20–36. Springer, 2022. 1, 2, 4

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6, 13

[7] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, pages 224–236, 2018. 2, 3, 4, 6, 7, 11

[8] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. Dkm: Dense kernelized feature matching for geometry estimation. In *CVPR*, pages 17765–17775, 2023. 1, 2

[9] Pierre Gleize, Weiyao Wang, and Matt Feiszli. Silk: Simple learned keypoints. In *ICCV*, pages 22499–22508, 2023. 2, 3, 4, 6, 7, 12

[10] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, pages 10–5244. Citeseer, 1988. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017. 3

[13] Menelaos Kanakis, Simon Maurer, Matteo Spallanzani, Ajad Chhatkuli, and Luc Van Gool. Zippypoint: Fast interest point detection, description, and matching through mixed precision discretization. In *CVPRW*, pages 6113–6122, 2023. 1, 2, 7

[14] A. Karpur, G. Perrotta, R. Martin-Brualla, H. Zhou, and A. Araujo. LFM-3D: Learnable Feature Matching Across Wide Baselines Using 3D Signals. In *3DV*, 2024. 2

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6, 11

[16] Viktor Larsson and contributors. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 6

[17] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, pages 2041–2050, 2018. 6, 11, 13

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 6, 11

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3

[20] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 2, 3, 6, 11, 12

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 2

[22] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, pages 6589–6598, 2020. 3, 4

[23] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *NeurIPS*, 30, 2017. 2

[24] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Repeatability is not enough: Learning affine regions via discriminability. In *ECCV*, pages 284–300, 2018. 2

[25] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. on Robotics.*, 33(5):1255–1262, 2017. 1, 2

[26] Erickson R Nascimento, Guilherme Potje, Renato Martins, Felipe Cadar, Mario FM Campos, and Ruzena Bajcsy. Geobit: A geodesic-based binary descriptor invariant to non-rigid deformations for rgb-d images. In *ICCV*, pages 10004–10012, 2019. 2

[27] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pages 3456–3465, 2017. 1

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 6

[29] Guilherme Potje, Gabriel Resende, Mario Campos, and Erickson R Nascimento. Towards an efficient 3d model estimation methodology for aerial and ground images. *Mach. Vis. and Applications.*, 28:937–952, 2017. 1

[30] Guilherme Potje, Renato Martins, Felipe Chamone, and Erickson Nascimento. Extracting deformation-aware local features by learning to deform. *NeurIPS*, 34:10759–10771, 2021. 2

[31] Guilherme Potje, Renato Martins, Felipe Cadar, and Erickson R Nascimento. Learning geodesic-aware local features from rgb-d images. *CVIU*, 219:103409, 2022. 2

[32] Guilherme Potje, Felipe Cadar, André Araujo, Renato Martins, and Erickson R Nascimento. Enhancing deformable local features by jointly learning to detect and describe keypoints. In *CVPR*, pages 1306–1315, 2023. 2

[33] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. *NeurIPS*, 2019. 2, 3, 5

[34] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571. Ieee, 2011. 2, 7

[35] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019. 1, 2, 7

[36] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020. 2, 3

[37] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, pages 8601–8610, 2018. 7

[38] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 1, 2

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 3

[40] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, pages 8922–8931, 2021. 1, 2, 4, 5, 6, 11, 12, 13

[41] Prune Truong, Martin Danelljan, Radu Timofte, and Luc Van Gool. Pdc-net+: Enhanced probabilistic dense correspondence network. *IEEE TPAMI*, 2023. 1, 2

[42] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *NeurIPS*, 33: 14254–14265, 2020. 2, 3, 4, 5, 6, 7, 12

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 2

[44] Changchang Wu. Towards linear-time incremental structure from motion. In *3DV*, pages 127–134, 2013. 1

[45] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018. 3

[46] Xiaoming Zhao, Xingming Wu, Jinyu Miao, Weihai Chen, Peter CY Chen, and Zhengguo Li. Alike: Accurate and lightweight keypoint detection and descriptor extraction. *IEEE TMM*, 2022. 1, 2, 3, 4, 5, 6, 7, 11

[47] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. Patch2pix: Epipolar-guided pixel-level correspondences. In *CVPR*, pages 4669–4678, 2021. 6, 12

[Supplementary Material]
XFeat: Accelerated Features for Lightweight Image Matching

In this supplementary material accompanying the main paper, we present a more detailed overview of the architecture of our proposed CNN backbone and the practices employed in the training process. Moreover, we provide an expanded set of qualitative results and extended discussion, providing additional contextualization with the current state-of-the-art methods. Code and weights are available at verlab.dcc.ufmg.br/descriptors/xfeat_cvpr24.

A. Backbone details

To maintain the backbone's structural simplicity, we employ a primary unit termed the basic layer. This unit is structured with a 2D convolution with square kernel sizes $k = 1$ or $k = 3$, complemented by ReLU activation and Batch Normalization. A stride of 2 in the convolution is applied for halving the spatial resolution as needed. The network's architecture is modular, comprising several basic layers as a basic block, as depicted in Fig. 6. Each block consists of two or three basic layers. The backbone of our network comprises six of these basic blocks, designed to halve the spatial resolution in each step while progressively augmenting the depth using the approach detailed in Sec. 3.1 of the main paper. The first basic layer on each block performs the spatial downsampling. Two additional basic blocks, in the end, are employed to perform the fusion of multi-resolution features and reliability map prediction, respectively. Preliminary experiments revealed that adding a single skip connection to the model as shown in Fig. 6 slightly increased performance, which has led to its incorporation in the final backbone design.

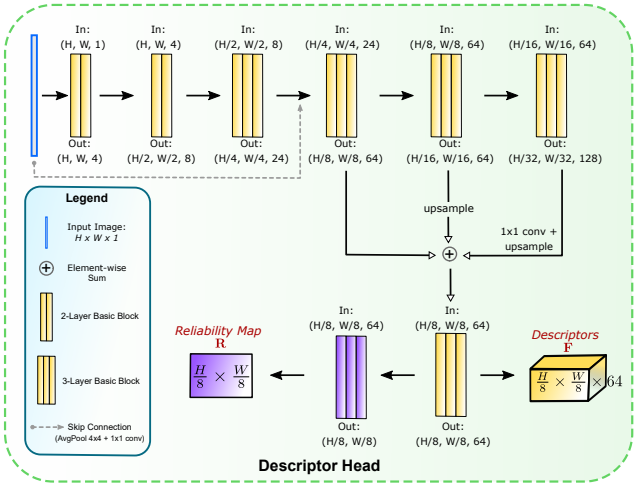

Figure 6. **Detailed descriptor backbone.** Our backbone is comprised of 23 convolutional layers, following the downsampling strategy described in Sec. 3.1 of the main paper. Our network is deeper compared to ALIKE [46] and SuperPoint [7] backbones in terms of layers, but due to the efficient downsampling strategy adopted, our network's inference is much faster.

B. Training description

We trained the network on a mix of Megadepth [17] scenes using the training split provided by [40] and synthetically warped pairs using raw images (without labels) from COCO [18] in the proportion of $6 : 4$ respectively. All image pairs were resized to $(W = 800, H = 600)$, and ground-truth correspondences were scaled accordingly. Our ablations show that hybrid training significantly improves generalization for small CNNs, as observed in high-capacity models [20]. The network was trained on batches of 10 image pairs using the Adam optimizer [15] with an initial learning rate of 3×10^{-4}, applying an exponential decay of 0.5 at every 30,000 gradient updates. Convergence is attained after 160,000 iterations, within 36 hours on a single NVIDIA RTX 4090 GPU, consuming 6.5 GB of VRAM in total, considering both training and synthetic warps done on the fly on GPU. Disk I/O is the predominant speed bottleneck due to the overhead of loading images and depth maps from the Megadepth dataset in their original resolution, which can be easily solved with a more careful data preparation scheme. The low memory usage of our method enables training on entry-level hardware, facilitating the fine-tuning or full training of our network for specific tasks and scene types.

C. Detailed timing analysis

This section reports a detailed timing analysis of our proposed solutions in sparse and semi-dense matching settings. Regarding XFeat*'s match refinement step, we show in Fig. 7 that the match refinement cost is negligible. More notably, even with the refinement step included, XFeat* achieves a similar matching time compared to XFeat with the same number of keypoints because refinement is performed after the nearest neighbor search. Additionally, we present the extraction running times for the most efficient methods available on an **Orange Pi Zero 3** equipped with a Cortex-A53 ARM processor. This device stands out as one of the **smallest and most affordable consumer-grade embedded computers** ($28). Considering its limited processing power, we adjusted the input resolution to 480×360

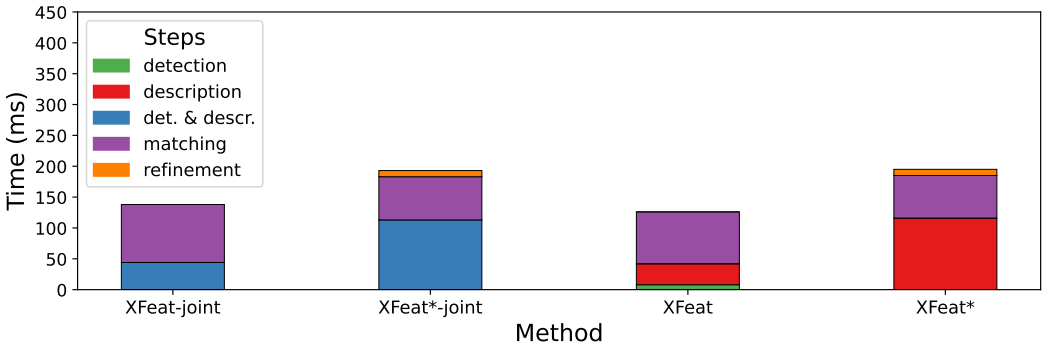

Figure 7. **Detailed timing analysis on i7-6700K CPU.** Required time by each step of our ablated methods.

for all methods and used their standard PyTorch implementation without any deployment optimization. Our findings show that XFeat operates at an average of 1.8 FPS, SuperPoint at 0.16 FPS, and ALIKE at 0.58 FPS, respectively. This experiment shows that XFeat is the only learned method capable of running over one FPS on a highly constrained embedded device that is not optimized for neural network inference.

D. Megadepth-1500 qualitative results

Fig. 8 shows more qualitative results of our two proposed approaches compared to the baseline methods used in the main paper. For more challenging cases such as strong viewpoint and illumination changes, XFeat and XFeat* exhibit exceptional robustness even compared to DISK [42] – the largest CNN architecture regarding floating point operations. We hypothesize that this robustness is attributed to our network's large receptive field and depth compared to shallower models such as SuperPoint, ALIKE, and SiLK [9], demonstrating the effectiveness of our featherweight backbone in the compute-accuracy trade-off.

E. ScanNet-1500 extended discussion

Recalling the results obtained in Tab. 2 of the main paper, XFeat and XFeat* surpass both fast and standard local feature extractors in pose accuracy while being significantly faster for indoor relative pose estimation. DISK and ALIKE, which were trained in the same Megadepth scenes as XFeat, display signs of overfitting in landmark imagery: they perform exceptionally well in strict thresholds (AUC@5°) on Megadepth-1500 test set, but their relative performance are similar or worse in tasks such as homography estimation and visual localization compared to XFeat and SuperPoint, as one can observe in Tab. 3 and Tab. 4 of the main paper.

We conjecture that XFeat produces less biased local descriptors due to our hybrid training with synthetic warps on COCO. SuperPoint also demonstrate increased generalization accross different downstream tasks and datasets due to its inherent self-supervised training strategy on synthetic warps. Hybrid training can encourage local feature representations to focus less on distinctive textures often present in landmark outdoor imagery that could bias the CNN training. In addition, the large receptive field of our network, as well as its increased layer depth compared to the other approaches, helps XFeat in indoor imagery (which often lacks distinctiveness at the local level), resulting in more consistent matches compared to DISK and ALIKE in ScanNet-1500, even though XFeat and the competitors were not trained on ScanNet data.

F. Comparison with learned matchers

Since XFeat* uses paired inputs when performing the refinement step, we provide additional comparisons of XFeat* (semi-dense matching) with popular learned matchers such as LoFTR [40] and LightGlue [20], and coarse-to-fine strategies as Patch2Pix [47], to elucidate the key differences. The results for these new approaches are shown in Tab. 6. Although XFeat* needs paired inputs for refinement, it fundamentally differs in its methodology from learned matchers, being only comparable to Patch2Pix, as we rely on traditional nearest neighbor search for matching, followed by a lightweight refinement of matches, incurring a negligible computational load (see Fig. 7). The requirement for paired inputs does not change the usual pipeline for SfM and visual localization tasks because XFeat*'s features can be stored for each image independently, as usually done for sparse settings. For instance, high-resolution feature maps are not required, unlike LoFTR, to produce refined matches.

Our techniques are, in fact, complementary to learned matchers; for example, LightGlue can be trained using both XFeat and XFeat* features. Learned matchers are more data hungry and much more expensive to train, *e.g.*, LoFTR uses 64 GPUs for 24 hours to be trained. XFeat*, for its turn, can be trained on a single 8 GB GPU. Furthermore, XFeat* offers up to 22× speedup over existing semi-

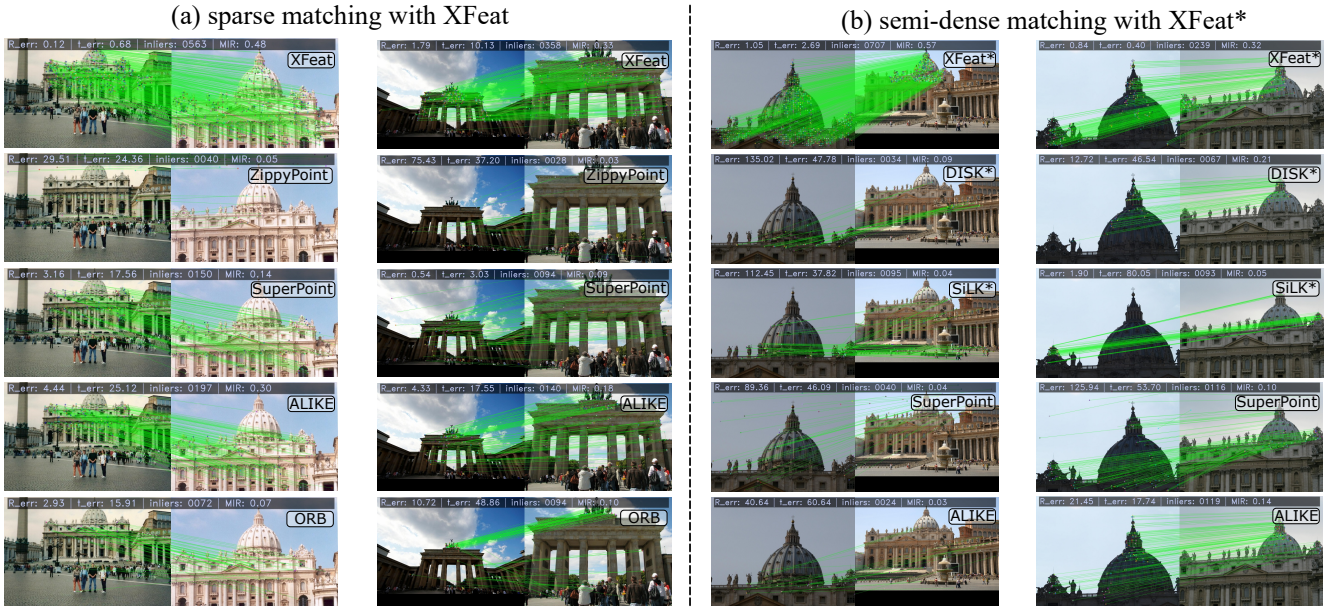

Figure 8. **Additional qualitative results on Megadepth-1500 [17, 40] landmark dataset.** XFeat and XFeat* are robust in demanding scenarios with significant viewpoint and illumination variations, outperforming even the more computationally intensive DISK model in semi-dense matching with 10,000 local features at a striking 16× speedup. In a sparse setting with 4,096 keypoints, our method, which is many times faster than ALIKE (5×) and SuperPoint (9×), demonstrates more robustness to wide baseline transformations due to the effective re-formulation of XFeat's backbone CNN.

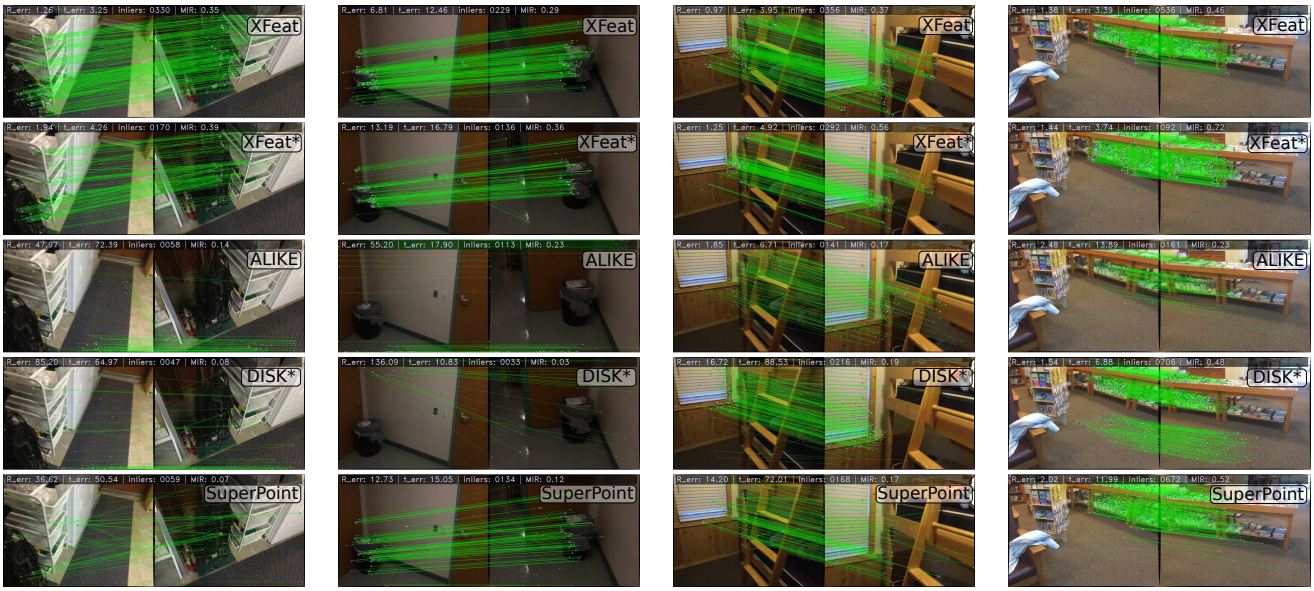

Figure 9. **Additional qualitative results on ScanNet-1500 [6, 40] indoor dataset.** Our proposed approaches consistently outperform state-of-the-art methods such as DISK and ALIKE in indoor imagery, both in terms of camera pose and inlier ratio. Notice that SuperPoint also often outperforms DISK and ALIKE. Appendix E provides a detailed discussion on the reasons behind our method's superiority.

dense solutions as shown in Tab. 6 and surpasses coarse-to-fine approaches such as Patch2Pix in accuracy, while being faster and delivering many more matches than sparse learned matchers as LightGlue. Naturally, XFeat, as a local descriptor, offers limited robustness to aggressive viewpoint changes and highly ambiguous image pairs compared to transformer-based feature matchers. Coupling a lightweight transformer such as LightGlue or LoFTR's linear trans-

Table 6. **Matchers comparison on Megadepth-1500.** Inference speed in pairs per second (PPS) @ 1,200 px. (i7-6700K CPU).

Method	**Type**	**AUC@5°**	**@10°**	**@20°**	**Acc@10°**	**MIR**	**#inliers**	**PPS**
LoFTR	learned matcher	68.3	80.0	88.0	93.9	0.93	3009	0.06
LightGlue	learned matcher	61.4	75.0	84.8	91.8	0.92	475	0.31
Patch2Pix	coarse-fine	47.8	61.0	71.0	77.8	0.59	536	0.05
XFeat*	coarse-fine	**50.2**	**65.4**	**77.1**	**85.1**	**0.74**	**1885**	**1.33**

former with XFeat's local features can open new directions in scalable, high-performance image matching tasks, facilitating advancements in both efficiency and accuracy that are pivotal for pushing the boundaries in visual navigation, augmented reality, and real-time visual SLAM.