Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 5131440, 15 pages
https://doi.org/10.1155/2023/5131440

Research Article

Feature Point Detection and Description Networks Based on Asymmetric Convolution and the Cross-Resolution Image-Matching Method

Ruixing Zhang,[1,2] Tao Yao ,[1,2] and Lianshan Yan[1,2,3]

[1]*School of Information and Electrical Engineering, Ludong University, Yantai 264025, China*
[2]*Yantai Research Institute of New Generation Information Technology, Southwest Jiaotong University, Yantai 264000, China*
[3]*School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610101, China*

Correspondence should be addressed to Tao Yao; yaotao@ldu.edu.cn

Received 21 September 2022; Revised 6 December 2022; Accepted 21 December 2022; Published 20 February 2023

Academic Editor: Mohammad R. Khosravi

Image matching can be transformed into the problem of feature point detection and matching of images. The current neural network methods have a weak detection effect on feature points and cannot extract enough sparse and uniform feature points. In order to improve the detection and description ability of feature points, this paper proposes a self-supervised feature point detection and description network based on asymmetric convolution: ACPoint. Specifically, first, feature point pseudolabels are learned from an unlabeled dataset, and pseudolabels are used for supervised learning; then, the learned model is used to update pseudolabels. Through multiple iterations of model training and label updating, high-quality labels and high-accuracy models are obtained adaptively. The asymmetric convolution feature point (ACPoint) network adopts an asymmetric convolution module to simultaneously train three convolution branches to learn more feature information, which uses two one-dimensional convolutions to enhance the backbone of square convolution from both horizontal and vertical directions and improve the representation of local features during inference. Based on the ACPoint network, a cross-resolution image-matching method is proposed. Experiments show that our proposed network model has higher localization accuracy and homography estimation ability on the HPatches dataset.

1. Introduction

The goal of image matching is to identify and align two images to match at the pixel level. The images to match are usually taken from similar scenes or targets and have a certain degree of compatibility [1, 2]. According to the statistics of Automated Imaging Association, more than 40% of visual perception applications rely on the accuracy and efficiency of image matching, including computer vision, image synthesis, remote sensing, military security, and medical diagnosis [3]. Image matching can be regarded as the detection and matching of image feature points. It is mainly divided into two parts: detecting feature points and descriptor vectors and using descriptors to match similar feature points in two images [4].

For images, features are specific structures, such as building edges, corners, and clearly shaped objects. These features are usually referred to as localized features, which usually need to be described by adjacent pixel blocks near the feature point position [5]. A novel hashing method is proposed [6], which combines the asymmetric hashing learning strategy and adaptive fuse multimodal features and learns binary codes as image features for efficiency. Features can be regarded as a simplified representation of the entire image. Using features for image matching reduces ineffective calculations, noise, and distortion. The descriptor vector

uniquely describes the feature points by recording the directional features and local appearance of feature points, such as appearance contours, and the description should have characteristics that are invariant to changes in illumination, translation, scale, and in-plane rotation [7].

In recent years, with the continuous development of convolutional neural networks, compared with traditional handcrafted features, neural networks can detect more sparse and uniform feature point sets from images, as well as feature descriptors with discriminative and matchable capabilities [8, 9]. At present, the development of neural network technology still relies on manually annotated datasets [10]. The semantics of dataset labels for object detection or image classification tasks is deterministic; however, the image feature points' concept is semantically ambiguous.

To address the lack of dataset labels, we use pseudolabel datasets for model training (see Figure 1). To make the feature point labels of the generated pseudoground-truth datasets more repeatable and accurate, we propose a self-supervised label solution based on confidence and label distance, called model adaptation technology (see Figure 1). The degree of association between labels generated by different models is proportional to confidence and inversely proportional to spatial distance. The model adaptation technology uses the two-dimensional distance and confidence to achieve low-density separation of feature points through the label data's distribution.

First, the pretrained model combined with homography adaptation technology is used to automatically label the feature point labels of the real image dataset [8], and then, the model adaptation technology is used to verify labels generated by different models to obtain the feature point labels with higher confidence. The comparison of feature points between different models can enhance the repeatability of feature points and make the generated labels have higher accuracy. Similarly, samples with high confidence will also improve the fitting ability of the model. Intramodel homography adaptation and crossmodel label comparison will help enhance the feature point detection capability of the model, as well as the feature point localization capability at lower resolutions.

The traditional VGG-style network structure is flat and lacks an effective feedback path, and it is difficult for the network model to achieve the accuracy of the complex multibranch structure [11–13]. The flat-style network has slightly lower accuracy, but the inference speed is very effective. Similarly, the complex multibranch structure makes the model difficult to implement and customize, reducing the inference speed and memory utilization [14, 15]. In order to combine the accuracy of the multibranch network and the inference speed of the flat network structure, this paper proposes an image feature point detection and description network based on asymmetric convolution: ACPoint. The network consists of a shared asymmetric convolutional encoder, feature point decoder, and descriptor decoder. The asymmetric convolution block (ACB) [14] of the encoder and decoder contains two one-dimensional convolutions and one square convolution and learns more feature information by simultaneously training three parallel branches. During inference, two one-dimensional convolutions are used to enhance the backbone of the square convolution from both horizontal and vertical directions, improving the representation ability of the square convolution for local features and inference speed by merging the three branches [13, 16].

We summarize our contributions as follows:

(1) We propose a feature point detection and description network based on asymmetric convolution to improve the accuracy of the model without increasing time complexity.

(2) We propose a self-supervised model adaptation method for benchmark label creation and improve label accuracy through continuous iterative updates.

(3) We propose a novel cross-resolution image-matching method based on the feature points and descriptors detected by the ACPoint network model.

2. Related Work

2.1. Image Matching. Image-matching methods are mainly divided into two types. Area-based methods use the entire image or a cropped image patch as a direct-matching target. The cross-correlation method and the correlation coefficient measurement method are used to align the image at the pixel level by minimizing the difference in image gray information [17, 18]. The Fourier transform method, the phase correlation method, the Walsh transform method, and other image-matching methods based on image domain transformation first transform the image information into the domain and then perform similarity matching on the image in the transformed domain [19–21]. Area-based image-matching methods are extremely sensitive to imaging conditions and image deformation (especially requiring extremely high overlap of image pairs) and have high computational complexity, which limits their application capabilities, and the most critical point is that the area-based matching method is only applicable to the same or similar scales and cannot solve the matching problem of cross-scale images.

Feature-based image-matching algorithms study the detection of physically significant structural features from images, including feature points, lines or edges, and salient morphological regions. The detected structural features are then matched, and a transformation function is estimated to align the rest of the images [22, 23]. For the entire feature-based matching framework, features can be regarded as a simplified representation of the entire image, which reduces ineffective computation and reduces the impact of noise, distortion, and other factors on matching performance.

There are currently two different approaches to feature-based matching: sparse matching by minimizing the alignment error and dense matching by finding the corresponding matching points for all points on the image [24]. Sparse matching relies on sparse feature points, and matching correspondence is obtained by filtering putative

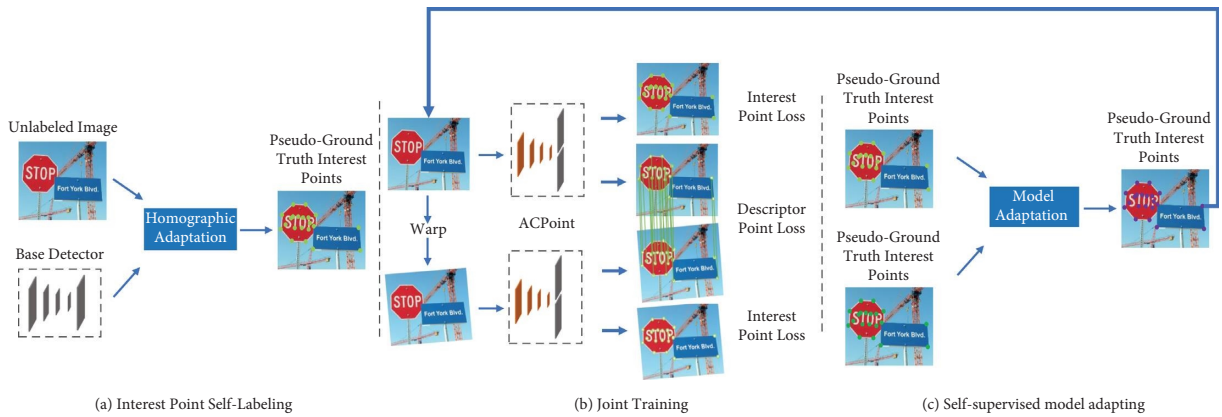


Figure 1: Self-supervised training overview.

matching pairs. Dense matching usually assumes that images are similar in the temporal domain, as in optical flow estimation of video sequences, and based on local smoothness assumptions [25]. Dense matching is difficult when image pairs are inconsistent in color or when there are a large number of repeating textureless regions. Compared with sparse matching, dense matching has stricter requirements for image pairs and is more computationally difficult.

2.2. Detector-Free Local Feature Matching. Detector-free methods remove the feature detector phase and directly generate dense feature matches. SIFT flow [26] is the first traditional detector-free dense matching method, which uses the optical flow method to realize the dense matching between two images from pixel to pixel. UCN [27] used the learning-based method for dense correspondence to directly extract the features from two images and perform the nearest neighbor search per pixel on the feature space to obtain the predicted match. NCNet [28] used an end-to-end dense matching method to obtain matching pairs by analyzing the neighborhood consistency of all possible corresponding points between a pair of images in a four-dimensional space. SuperGlue [29] used a learning-based local feature-matching method, which uses the graph neural network (GNN) to learn feature point matching. LoFTR [25] used CNN to treat every pixel as feature points to extract dense features and used the transform's global receptive field to obtain dense matching of low-texture regions. In these works, dense matching is affected by receptive fields, and correspondences generated by neighboring regions lack sufficient robustness. Dense matching would incur huge computational costs and rely on more complex models.

2.3. Detector-Based Local Feature Matching. Classic keypoint detectors, such as SIFT [30] and SURF [31], use the histogram of oriented gradients (HOGs) as the descriptor to maximize detection accuracy. SIFT can reliably identify objects even in the presence of noise and partial occlusion, but its HOG-based descriptors need to calculate intensity gradients, resulting in low computational speed and unfavorable real-time applications. SURF is optimized for speed and is still too computationally expensive. In addition, some binary-based descriptors such as ORB, FREAK, and KAZE rely on the intensity information of the image itself, encode the intensity information around keypoints as a string of binary numbers, and utilize binary-distinguishing features [32–34]. Traditional methods lack the description of global information, and pixel-by-pixel detection is prone to feature point aggregation, and cluttered and dense feature points will increase the difficulty of later matching.

The fast [35] corner detector is the first algorithm to address fast corner detection as a machine learning problem. Close to traditional patch-based detection and description methods, LIFT [36] employs sliding-window detection similar to SIFT and is the first end-to-end pipeline but still requires the supervision of ground truth generated by classical SIFT and SFM. Dosovitskiy et al. [37] proposed a general feature detection method using unlabeled data to train convolutional neural networks. Yang et al. [38] proposed a nonrigid registration method based on the same idea, where they used a pretrained VGG network layer to generate a multiscale feature descriptor while preserving convolutional information and local features. Simo-Serra et al. [39] used a Siamese network to focus on training samples that were difficult to distinguish categories and input image patch pairs and used the nonlinear mapping output by CNN as a descriptor and Euclidean distance to calculate similarity and minimize its hinge loss. The TILDE [40] interest point detection system used a principle similar to homographic adaptation; however, this approach does not benefit from the power of large fully convolutional neural networks. Superpoint [8] used a self-supervised pipeline to train detectors and descriptors simultaneously and outperformed traditional algorithms in HPatches [41] evaluation using homography adaptation techniques. These features or descriptors outperform hand-crafted descriptors on geometric matching tasks. However, the feature points extracted by the existing neural network models are still not sufficient and accurate. These differences are summarized in Table 1.

Table 1: Qualitative comparison to relevant methods.

	Interest points	Descriptors	Full image input	Single network	Self-supervised	Real time
ACPoint (ours)	√	√	√	√	√	√
LoFTR			√	√		
LIFT	√	√				
UCN		√	√	√		
TILDE	√			√		
DeepDesc		√		√		
SIFT	√	√				
ORB	√	√				√

3. Method

3.1. Asymmetric Convolution. Asymmetric convolution is used for model and parameter compression by approximating square convolution. Some previous works have shown that a conventional $n \times n$ convolution can be split into an $n \times 1$ convolution and a $1 \times n$ convolution [16], decomposing the square convolution can get more decoupled features while reducing the number of parameters and speeding up the training of the network.

ACNet [14] discovered a property of asymmetric convolutions: multiple size compatible 2D convolutions share the same sliding window with the same stride to perform linear operations on the same input, resulting in outputs of the same resolution. When these convolution kernels are added at corresponding positions, the obtained fused convolution kernel produces the same convolution result as follows:

$$I \times K^1 + I \times K^2 = I \times \left(K^1 \oplus K^2\right), \tag{1}$$

where I is the input feature rectangle, K^1 and K^2 are the two linear convolution kernels, and $\oplus$ is the element-wise addition of the linear convolution kernels at the corresponding positions. Affected by the shape of the linear convolution kernel, the matrix I needs to be edge clipped or filled during the convolution process.

3.2. Reparameterization

3.2.1. BN Fusion. Batch normalization (BN) can accelerate the convergence speed of the network, making network training easier [42]. At present, many deep models use the BN layer for batch normalization after the convolution layer to improve the generalization ability of the model. During the training process, the BN layer learns the mean μ and variance σ^2 of all elements x_i in a minibatch of input features and then subtracts the mean and divides the standard deviation from input elements. Finally, affine transformation is carried out with the learnable parameters γ and β to realize translation and scaling.

After training, the parameters of the convolution kernel and the BN layer are fixed, and the BN layer's parameters are represented by the following formula:

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i, \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m} \left(x_i - \mu_B\right)^2, \\ \widehat{x}_i &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \\ \mathrm{BN}_{\gamma,\beta}\left(x_i\right) &= \gamma \widehat{x}_i + \beta. \end{aligned} \tag{2}$$

We bring the formula $y = \omega \bullet x + b$ of the convolutional layer into the BN layer as follows:

$$\mathrm{BN}_{\gamma,\beta}\left(x_i\right) = \gamma \frac{\omega \bullet x + b - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta = \frac{\gamma \bullet \omega}{\sqrt{\sigma_B^2 + \epsilon}} \bullet x + \frac{\gamma}{\sqrt{\sigma_B^2 + \epsilon}} \bullet \left(b - \mu_B\right) + \beta. \tag{3}$$

Let $\widehat{\omega} = (\gamma \bullet \omega / \sqrt{\sigma_B^2 + \epsilon}), \widehat{b} = (\gamma / \sqrt{\sigma_B^2 + \epsilon}) \bullet (b - \mu_B) + \beta$; then, $\mathrm{BN}_{\gamma,\beta}(x_i) = \widehat{\omega} \bullet x_i + \widehat{b}$. The homogeneity of convolution allows equivalent fusion of the BN layer's parameters into the convolutional layer with bias, resulting in a new convolution kernel and bias term.

3.2.2. ACB Fusion. The feature map (M) with C channels output by the convolution kernel (F) is expressed as

$$O_{:,:,j} = \sum_{k=1}^{C} M_{:,:,k} \times F_{:,:,k}^{(j)}. \tag{4}$$

In the convolutional neural network (CNN), in order to suppress the overfitting of the network model and accelerate the network convergence speed, the BN layer will be added after the linear transformation to enhance the feature expression ability of the model. After the convolutional layer and the BN layer are fused, Equation (4) can be expressed as

$$O_{:,:,j} = \left(\sum_{k=1}^{C} M_{:,:,k} \times F^{(j)}_{:,:,k} - \mu_j \right) \frac{\gamma_j}{\sigma_j} + \beta_j, \tag{5}$$

where μ_j and σ_j are the values of the channel-wise mean and standard deviation of batch normalization and γ_j and β_j are the learned scaling factor and bias term, respectively.

$$\begin{aligned} F'^{(j)} &= \frac{\gamma_j}{\sigma_j} F^{(j)} \oplus \frac{\overline{\gamma}_j}{\overline{\sigma}_j} \overline{F}^{(j)} \oplus \frac{\widehat{\gamma}_j}{\widehat{\sigma}_j} \widehat{F}^{(j)}, \\ b_j &= -\frac{\mu_j \gamma_j}{\sigma_j} - \frac{\overline{\mu}_j \overline{\gamma}_j}{\overline{\sigma}_j} - \frac{\widehat{\mu}_j \widehat{\gamma}_j}{\widehat{\sigma}_j} + \beta_j + \overline{\beta}_j + \widehat{\beta}_j, \\ O_{:,:,j} + \overline{O}_{:,:,j} + \widehat{O}_{:,:,j} &= \sum_{k=1}^{C} M_{:,:,k} \times F'^{(j)}_{:,:,k} + b_j, \end{aligned} \tag{6}$$

where F' is the convolution kernel after fusion, b_j is the bias term, and $O_{:,:,j}$, $\overline{O}_{:,:,j}$, and $\widehat{O}_{:,:,j}$ are the outputs of the original branch.

3.3. Label Generation and Validation

3.3.1. Homography Adaptation. The homography adaptation technique imitates the change in the camera angle of view to perform random homography transformation on the real image. In order to well simulate the homography of camera transformation, the homography adaptation technique uses a truncated normal distribution to sample within a predetermined range of translation, scaling, in-plane rotation, and symmetric perspective distortion.

$$H(I; f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} H_i^{-1} f_\theta(H_i(I)), \tag{7}$$

where $f_\theta(\bullet)$ represents the initial interest point function we wish to adapt, I is the input image, H is a random homography, and N_h is the number of homographic warps.

The homography-transformed image is sent to the model to detect feature points, and then, the detected feature points are restored to original image coordinates. The confidence of the detected feature points is averaged, and then, the final feature point coordinates are filtered out by threshold.

3.3.2. Model Adaptation. For labels generated by different models for the same image, each label has coordinate information and confidence. The higher the confidence, the higher the probability that the point is a feature point. We perform label selection on the dataset based on label confidence and spatial distance as follows:

$$\mathrm{Corr}\left(m_i, \widehat{m}_j\right) = \min_{j \in 1,\ldots,N} \left(1 - \frac{p_{\widehat{m}_j}}{\sum_{n=1}^{N} p_{\widehat{m}_n}} \right) \left\| m_i - \widehat{m}_j \right\|, \tag{8}$$

where $\|m_i - \widehat{m}_j\| < \varepsilon$, ε is set to 3, which limits the coordinate error of the corresponding point to 3 pixels. The association degree between feature point pairs is proportional to confidence and inversely proportional to the spatial distance. The smaller the distance measure, the higher the degree of association. When there are multiple corresponding feature points within the error range, the point with the smallest distance metric is selected as the verification label point, and the points where m_i and $\widehat{m}_j$ are verified by each other should be reserved as feature points.

3.4. Focal Ratio. The ratio of the focal lengths between the images is calculated from the input images I_1 and I_2. ACPoint is used to calculate the feature points of the image pairs I_1 and I_2, and the feature correspondence and homography relationship between the images are obtained after matching. The image I_2 is mapped to $I_{2'}$ according to the homography matrix H, and the image area of $I_{2'}$ can be approximately regarded as a convex quadrilateral. Calculate the area of the convex quadrilateral according to the image vertices of $I_{2'}$ and obtain the ratio of the focal length by comparing the areas of I_2 and $I_{2'}$ as follows:

$$\begin{aligned} \mathrm{Area}_{\overline{L}} = \frac{1}{2} \sum_{i=1}^{n} \Big(& P_x[(i+1) \bmod (n)] P_y[i] \\ & - P_x[i] P_y[(i+1) \bmod (n)] \Big), \end{aligned} \tag{9}$$

where $\mathrm{Area}_{\overline{L}}$ represents the area of I_2 projected to the corresponding position in I_1, Area_L represents the actual area of I_2, P is the polygon vertex matrix stored in clockwise order, $P_x[i]$ and $P_y[i]$ are, respectively, the ith vertex abscissa and ordinate, and the number of vertices n is 4.

The final focal length ratio of the global camera to the local camera is

$$F_{\mathrm{ratio}} = \frac{\mathrm{Area}_L}{\mathrm{Area}_{\overline{L}}}. \tag{10}$$

4. ACPoint Architecture

4.1. Shared Encoder. As shown in Figure 2, our model adopts a VGG-style encoder to extract semantic features and reduce the dimensionality of images [12]. The shared encoder uses different modules for training and inference, respectively. The ACB module shown in Figure 3 is used to enrich the feature space during training, and the flat 3×3 convolutional module is replaced during inference. The model uses the ELU (exponential linear unit) [43] activation function to increase the nonlinearity of the network and then uses parallel maximum pooling and average pooling layers to reduce the image dimension.

The role of the encoder is to compress the input image into a latent spatial representation, which maps the input image $I \in \mathbb{R}^{H \times W}$ into an intermediate tensor $\mathscr{B} \in \mathbb{R}^{H_c \times W_c \times F}$

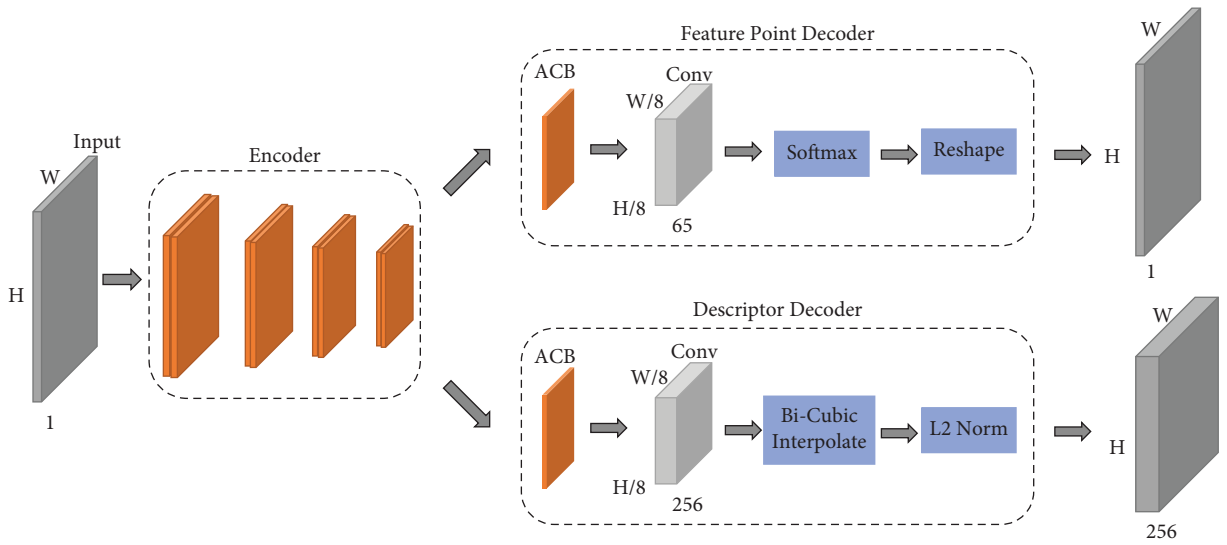


Figure 2: ACPoint network structure.

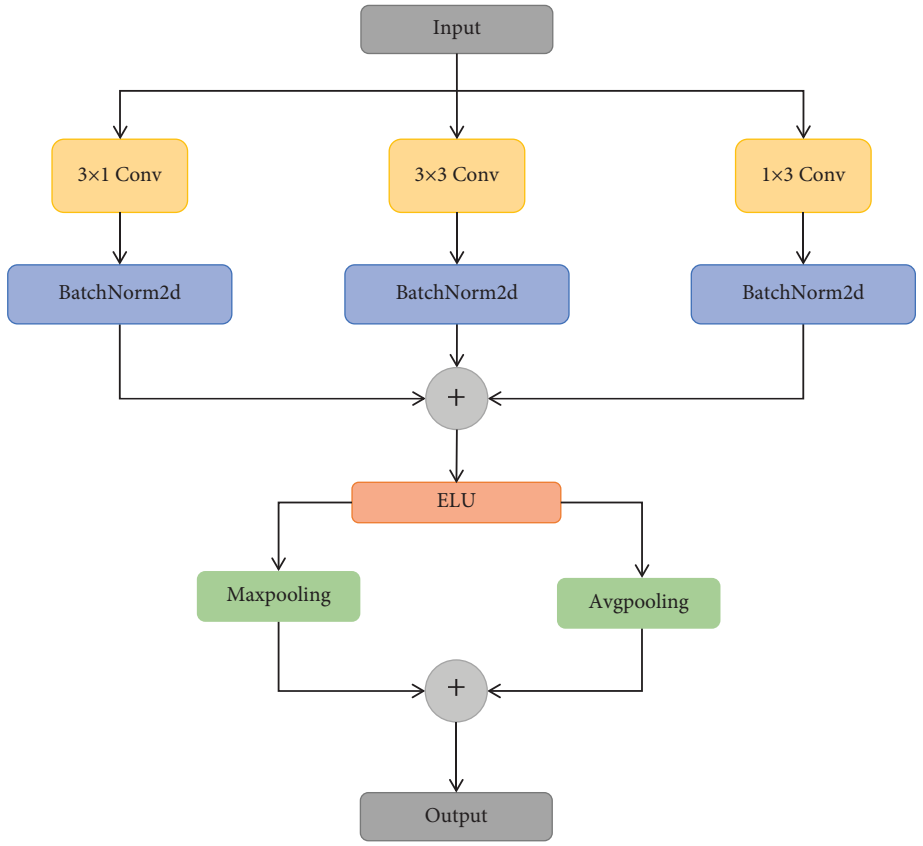


Figure 3: Asymmetric convolution block.

with smaller dimensions and larger channel depth so that the neural network can learn the most informative features. We integrate the pixels of the 8×8 region on the input image into one unit on the low-latitude output by three 2×2 pooling operations, reducing the $H \times W$ input image to $H_c = H/8$ and $W_c = W/8$.

4.2. Feature Point Decoder. Each pixel value output by using the feature point decoder corresponds to the probability that the pixel on the input image belongs to the feature point. Feature point detectors with explicit decoders use pixel shuffle to upsample feature maps back to full resolution size. The feature point detector head computes $X \in \mathbb{R}^{H_c \times W_c \times 65}$

and outputs a tensor of size $\mathbb{R}^{H\times W}$. 65 output channels, respectively, correspond to the 8×8 pixel area on the input image, and the remaining channel represents that there are no feature points in this area.

4.3. Descriptor Decoder. The descriptor decoder computes $D \in \mathbb{R}^{H_c\times W_c\times 256}$ and outputs a tensor of size $\mathbb{R}^{H\times W\times D}$. First, the decoder outputs a semidense grid of descriptors and performs pixel-wise patch normalization in the feature space to output a descriptor. Then, we perform bicubic interpolation of the descriptor and obtain the weighted average of the sixteen nearest samples in each direction at that location. Finally, L2 normalization is performed on unit length, and the descriptor corresponding to the feature point is obtained.

4.4. Loss Functions. The final loss function consists of two parts: loss L_p for the feature point detector and L_d loss for the descriptor detector. During the training process, for a given input image, the homography ground truth H is first randomly generated, and H is used to generate the corresponding warped image and the pseudoground-truth feature point label of the warped image. We use pairs of original and synthetic warped images to optimize both parts of the loss at the same time, and the final loss is as follows:

$$L\left(X, X', D, D', Y, Y', S\right) = L_p(X, Y) + L_p\left(X', Y'\right) + L_d\left(D, D', S\right). \tag{11}$$

The feature point loss L_p is the fully convolutional cross-entropy loss over the unit $x_{hw} \in X$, and we call the true feature point labels Y and the independent matrix elements y_{hw}. The feature point loss function is

$$L_p(X, Y) = \frac{\sum_{\hbar=1,w=1}^{H_c,W_c} l_p\left(x_{\hbar w}, y_{\hbar w}\right)}{H_c W_c}, \tag{12}$$

where lp denotes as follows:

$$l_p\left(x_{\hbar w}, y\right) = -\log\left(\frac{\exp\left(x_{\hbar wy}\right)}{\sum_{k=1}^{65}\exp\left(x_{\hbar wk}\right)}\right). \tag{13}$$

The descriptor loss is applied to all pairs of descriptor units, $d_{hw} \in D$ from the input image and $d'_{h'w'} \in D'$ from the warped image. The induced homography correspondence between descriptor units (h, w) and (h', w') can be written as

$$S_{hwh'w'} = \begin{cases} 1, & \left\|\widehat{Hp_{hw}} - p_{h'w'}\right\| \le 8, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

where p_{hw} represents the position of the center pixel in the cell (h, w) and $\widehat{Hp_{hw}}$ represents the cell position p_{hw} multiplied by the homography H and divided by the last coordinate. This is typically used to convert homogeneous coordinates back to Euclidean coordinates. We use S to denote the entire corresponding set of a pair of images. We use the hinge loss with the positive margin m_p and negative margin m_n and use the sparse loss to reduce the computational cost of the training process. The descriptor loss is defined as

$$L_d\left(D, D', S\right) = \frac{\sum_{h=1,w=1}^{H_cW_c} l_d\left(d_{hw}, d'_{h'w'}, S_{hwh'w}\right)}{N_{\text{Matc}\hbar}} + \frac{\sum_{h'=1,w'=1}^{H_cW_c} l_d\left(d_{hw}, d'_{h'w'}, S_{hwh'w'}\right)}{M_{\text{Nomatc}\hbar}}, \tag{15}$$

where $l_d(d, d', s) = s \times \max(0, m_p - d^T d') + (1 - s) \times \max(0, d^T d' - m_n)$, $m_p = 1$, and $m_n = 0.2$.

5. Experimental Details

In this section, we provide some implementation details for training the ACPoint model. The ACPoint network consists of a shared asymmetric convolutional encoder, feature point decoder, and descriptor decoder. The asymmetric convolutional encoder adopts a VGG-style network structure with 8 asymmetric convolutional blocks (ACB) of size 64-64-64-64-128-128-128-128. As shown in Figure 2, the ACB module adopts three branches of 3×3, 3×1, and 1×3 to learn feature information at the same time, and each branch is followed by a BN layer for batch normalization. After every two layers of the ACB module, the parallel maximum pooling layer and the average pooling layer are used to reduce the image dimension, and the window size and the stride size of the pooling layer are 2.

The pooling operation is a commonly used downsampling operation, which can reduce the parameter matrix's size and the feature map's dimension, reduce the parameters and calculation amount of the model, and effectively prevent overfitting. The pooling operation continuously abstracts the regional features of the feature map, which increases the translation invariance to a certain extent, but pooling also inevitably loses information. Average pooling averages local values, which are biased towards the overall characteristics of the background. It retains the

overall feature information of the feature map but easily loses details. Maximum pooling is to take the maximum value of the local area, and it is biased towards features such as texture outlines, which can filter out more useless information, and is sensitive to edge gradients. It is easy to select features with higher recognizability and better retention of texture information. The parallel connection method of average pooling and maximum pooling loses less information than single pooling so that information can be transmitted better.

The decoder reconstructs the input from the latent representation space, and both the feature point decoder head and the descriptor decoder head have a 256-dimensional ACB module, followed by a 1×1 convolutional layer. The interest point detector has 65 dimensions, and the descriptor detector has 256 dimensions. All convolution modules in the network are followed by an ELU activation function. Compared with the RELU activation function, the gradient of ELU is nonzero for all negative values, there is no problem of neuron death, and as a nonsaturating activation function, it will not encounter the situation of gradient explosion or disappearance. It is continuous and differentiable at all points. The ELU activation function is used to shorten neural network training time and improve accuracy.

We adopt MS COCO 2017 [10] as our real image dataset and use the Superpoint [8] and DeepFEPE [44] pretrained feature point detection models to generate pseudoground-truth datasets, respectively. The images were converted to grayscale images and kept the original resolution, while the images were subjected to 100 homography transformations and model detection to create initial feature point labels. For the 8×8 pixel blocks on the input image, of the 65 channels of the feature heatmap obtained by model detection, one channel represents whether there are feature points and the remaining 64 channels represent the probability of each pixel point. The feature point decoder uses pixel shuffle to sample the full-resolution size image on the feature heatmap. When reshaping back to the original size, the point with the highest score is retained by softmax as the quasi-feature point, and the scores of the remaining positions are zeroed out; only the quasi-feature points and their probabilities are retained. In order to increase the applicability of the model, 100 random homography transformations were carried out, the 100 heatmaps of not identical feature points were superimposed and normalized, and then, the quasi-feature points below the threshold were removed. We set the threshold to 0.015.

In order to make the feature points detected by the model sparse and uniform, nonmaximum suppression (NMS) is used to suppress elements that are not maximal in the local range. We take the NMS value to be 4 to ensure that each feature point has no other feature points within the 9×9 range centered on itself. Then, we use model adaptation technology to compare the labels generated by different models to obtain more accurate feature point labels. These labels are used as benchmark labels to perform supervised learning on the network, the trained model is combined with model adaptation technology to construct new feature point labels, and the accuracy of the labels is improved through continuous iteration.

In order to improve the robustness of the network for illumination and perspective changes during training, standard data augmentation techniques such as random Gaussian noise, motion blur, and brightness adjustment are also used. First, we used a grid search for the required parameter combinations and the 5-fold cross-validation method to fit them onto toy experiments with small sample sets. Finally, the best combination was selected according to the cross-validation scores. In addition, the AdamW stochastic gradient descent optimization algorithm was adopted. The learning rate is automatically adjusted by the adaptive mechanism, the model training process basically requires no intervention, and the hyperparameters are well interpretable. All training is performed based on the PyTorch framework with a minibatch size of 16 and the ADAMW solver with parameters $l_r = 0.0001$ and $\beta = (0.9, 0.999)$.

6. Experiment

In this paper, we use the MS COCO 2017 image dataset to train the ACPoint network model and the HPatches dataset to test the accuracy of homography estimation. The HPatches dataset includes 57 illumination scenes and 59 viewpoint scenes, with a total of 696 individual images grouped into 580 image pairs. The repeatability of feature points refers to the probability that the feature points detected in the first image also appear in the second image. We use the repeatability of detection points on image pairs to test the model's ability to detect feature points. Table 2 shows the feature point repeatability of different detectors in different scenes, and our model has better performance under illumination and viewing angle changes.

The total number of parameters to train is 1.3 million. The model occupies less than 5 MB of memory space. We take a 240×320 image as an example to analyze the calculation cost, and the floating point arithmetic is about 6.5GFLOPs. We use AMD Ryzen 7 5800H and NVIDIA GeForce RTX 3060 hardware devices with Python 3.6 and PyTorch 1.10 to test the running time on the HPatches dataset. The image input size is 240×320. Superpoint can reach 18.71FPS, DeepFEPE can reach 15.67FPS, and our proposed ACPoint network can reach 17.84FPS, about 4.8% more running time than that of Superpoint; however, the performance is greatly improved. The repeatability of feature points increases by 6% in illumination scenes and increases by about 9.8% in viewpoint scenes.

As shown in Tables 3 and 4, we comprehensively evaluate the model performance from three aspects: homography estimation, detector metrics, and descriptor metrics. Homography estimation first calculates the transformation matrix between images according to the feature point correspondence of the image and compares the average accuracy of feature point detection with the label homography matrix to measure the algorithm's ability to estimate image homography. The more accurate the homography estimation, the more accurate the image matching. ε is the

Table 2: HPatches detector repeatability.

	57 illumination scenes		59 viewpoint scenes	
	NMS = 4	NMS = 8	NMS = 4	NMS = 8
Random	0.101	0.103	0.100	0.104
Harris	0.620	0.533	0.556	0.461
Shi	0.606	0.511	0.552	0.453
FAST	0.575	0.472	0.503	0.404
Magicpoint	0.575	0.507	0.322	0.260
Superpoint	0.652	0.631	0.503	0.484
ACPoint (ours)	**0.712**	**0.656**	**0.602**	**0.528**

The meanings in bold in Table 2 represent the values of the best performing methods for the indicators in this column. Meanwhile, the method proposed in this paper has the best performance in all indexes in this table.

Table 3: HPatches homography estimation.

	Homography estimation		
	$\varepsilon = 3\uparrow$	$\varepsilon = 5\uparrow$	$\varepsilon = 10\uparrow$
SIFT	0.750	0.802	0.824
SURF	0.614	0.700	0.755
ORB	0.455	0.524	0.566
LIFT	0.689	0.698	0.722
LoFTR	0.659	0.756	0.846
Superpoint	0.753	0.821	0.845
DeepFEPE	0.752	0.822	0.851
ACPoint (ours)	**0.778**	**0.845**	**0.871**

The meanings in bold in Table 3 represent the values of the best performing methods for the indicators in this column. Meanwhile, the method proposed in this paper has the best performance in all indexes in this table.

Table 4: HPatches detector and descriptor performance.

	Detector metrics		Descriptor metrics	
	Rep.↑	MLE.↓	NNmAP↑	M.score↑
SIFT	0.512	1.163	0.695	0.267
SURF	0.510	1.363	0.700	0.265
ORB	**0.711**	1.068	0.455	0.151
LIFT	0.434	1.137	0.665	0.264
Superpoint	0.606	1.156	0.833	**0.545**
DeepFEPE	0.632	1.071	0.780	0.449
ACPoint (ours)	0.656	**1.024**	**0.845**	0.534

error threshold for determining the detected position point relative to a set of real feature points; that is, the error distance at which the pixel point is judged has to be correct.

Repeatability (Rep) tests the ability of the model to detect feature points. The higher the repeatability is, the more potential feature points will correspond. The matching location error (MLE) is used to calculate the correctly detected feature point location error, and the value range is $(0, \varepsilon)$, where ε is 3. The nearest neighbor mean precision (NNmAP) computes the distinguishability of descriptors by measuring the area under the curve of the precision-recall (PR) curve using nearest neighbor matching. The discriminative ability of descriptors is evaluated by multiple descriptor distance thresholds, calculated symmetrically over image pairs, and averaged. The matching score (M.s) measures the overall performance of the feature point detector and descriptor combination by measuring the ratio of the ground-truth correspondences recovered by the algorithm to the number of features detected in the shared viewpoint region. It is also calculated symmetrically over image pairs and averaged.

The linear convolutional branch in the ACB module enhances the model's extraction ability of the feature point and the single stress estimation ability of the model. The detection of ORB tends to form sparse clusters of feature points in the image, which can achieve the highest repeatability, but too sparse feature clusters also lead to difficulty in image matching. NMS is used by the model to be sparsely processed to extract feature points, making the final characteristic point sparse and uniform, and the reduction in the number of feature points will cause repetitive reductions. Superpoint scores well on descriptor-centric metrics, but optimization of the matching score does not lead to better matching or further homography estimation.

In order to obtain sparse, uniform, and accurate feature points, we use the sparse loss instead of the dense loss when training the descriptor loss, which can speed up the training time but also make M.score slightly lower than Superpoint. Benefiting from the enhanced detection capability of asymmetric convolutions for feature points, our model outperforms other methods on homography estimation, average nearest neighbor accuracy, and matching localization error.

As shown in Table 5, the ablation experiments are used to verify the role of each part of the network model. Because each part of the model is an essential part of the network, we use the most commonly used modules as the baseline and replace the corresponding module to verify its effectiveness. The convolutional layer, RELU activation function, and max pooling layer are used as the baseline to evaluate performance. It can be seen that the experimental results show that the asymmetric convolutional modules, ELU activation functions, and mixed pooling layers are used separately to have a small performance improvement to the model, which proves that each module is effective for the model. In addition, the combination strategy of each module enables the model to achieve optimal performance.

We iteratively update pseudolabels through model adaptation technology to continuously improve the accuracy and quantity of labels, which help improve model accuracy. After iteration, the number of labels increased by 29.86%. The experiments in Figure 4 show that the iterative labels have lower matching positioning accuracy, higher nearest neighbor average accuracy and matching score, and better accuracy in homography estimation. A slight decrease in repeatability indicates that the model also filters out some feature point correspondences that are irrelevant to matching, making the final feature point correspondences more accurate throughout the image, as shown in Figure 5.

As shown in Figure 6, we adopt a 5-fold cross-validation method to prove the stability of the model. 25,000 images are randomly extracted from the final data set and divided into 5 parts. The four parts are used as a training set, and the remaining part is used as a validation set for 5 training sets. The experimental results prove that our model has strong stability and generalization capabilities.

Table 5: HPatches ablation experiment.

	Homography estimation			Detector metrics		Descriptor metrics	
	$\varepsilon = 1\uparrow$	$\varepsilon = 3\uparrow$	$\varepsilon = 5\uparrow$	Rep.↑	MLE.↓	NNmAP↑	M.score↑
Conv + RELU + max pooling	0.486	0.764	0.821	0.644	1.093	0.800	0.490
ACB + RELU + max pooling	0.495	0.775	0.830	0.647	1.074	0.806	0.491
Conv + ELU + max pooling	0.497	0.771	0.824	0.645	1.091	0.811	0.490
Conv + RELU + mix pooling	0.500	0.765	0.827	0.648	1.072	0.809	0.500
ACB + ELU + mix pooling	**0.548**	**0.778**	**0.845**	**0.656**	**1.024**	**0.845**	**0.534**

The meanings in bold in Table 5 represent the values of the best performing methods for the indicators in this column. Meanwhile, the method proposed in this paper has the best performance in all indexes in this table.

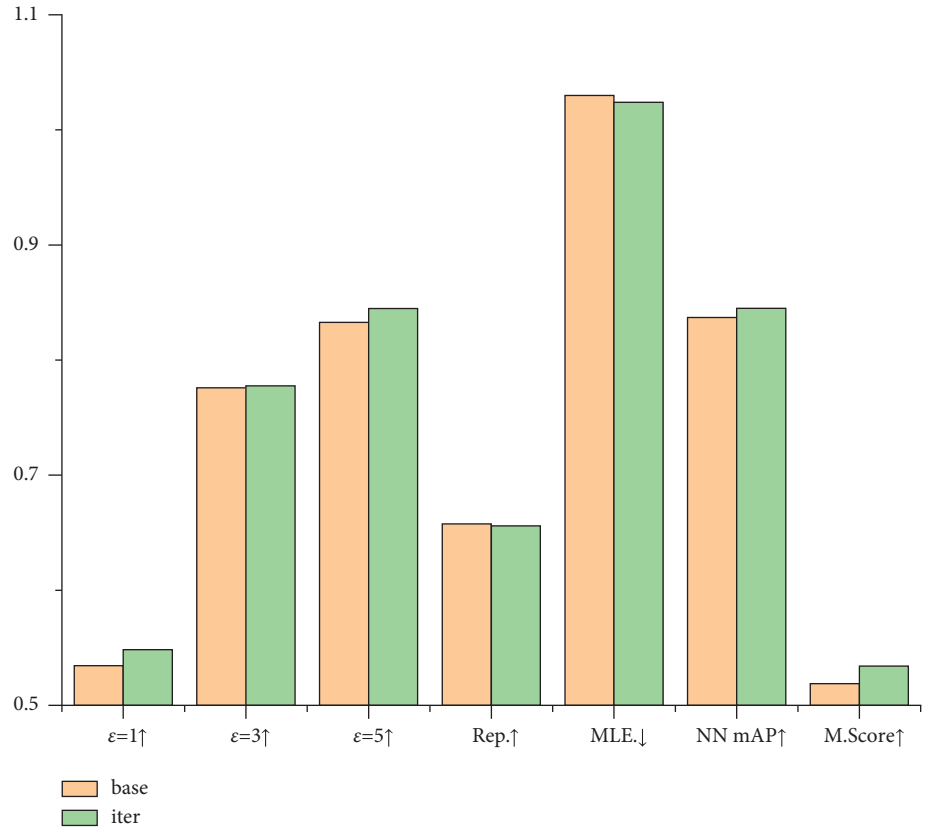


Figure 4: Metric comparison after iteration.

As shown in Figure 7, traditional feature point detectors such as SIFT and SURF detect a large number of potential feature points that are densely clustered and susceptible to noise. At the same time, it is easy to miss some points whose external characteristics are not obvious. The feature points detected by the traditional methods are many and messy, which will greatly increase the difficulty of matching in the later stage. Too many points bring a huge computational and storage burden to feature point matching, and sparse and uniformly accurate points are necessary. The detection of feature points by Superpoint and DeepFEPE is sparse and uniform, but they still lack sufficient detection ability to detect potential feature points as accurately as possible. Our model has higher feature detection ability, and the detected feature points are accurate and uniform.

As shown in Figure 8, we tested the feature point matching of the algorithm on images. Although the traditional algorithm detected a huge group of feature points, the final matching effect was poor. In the deep method, our proposed model can not only detect sparse, uniform, and accurate feature points but also achieve uniform and accurate feature matching on remote sensing images.

We use a zoom camera to capture images from different perspectives of the same scene. To verify the effect of image matching across focal lengths, matching image blocks are taken from a local image with a resolution of 4936 × 3266, the original global image has a resolution of 4936 × 3266, and the resolution is adjusted to 600 × 397 when the difference is 8 times.

We use ACPoint to detect feature points in the image, calculate the focal length scale of the image based on the feature points of the image, and help the image pair reach

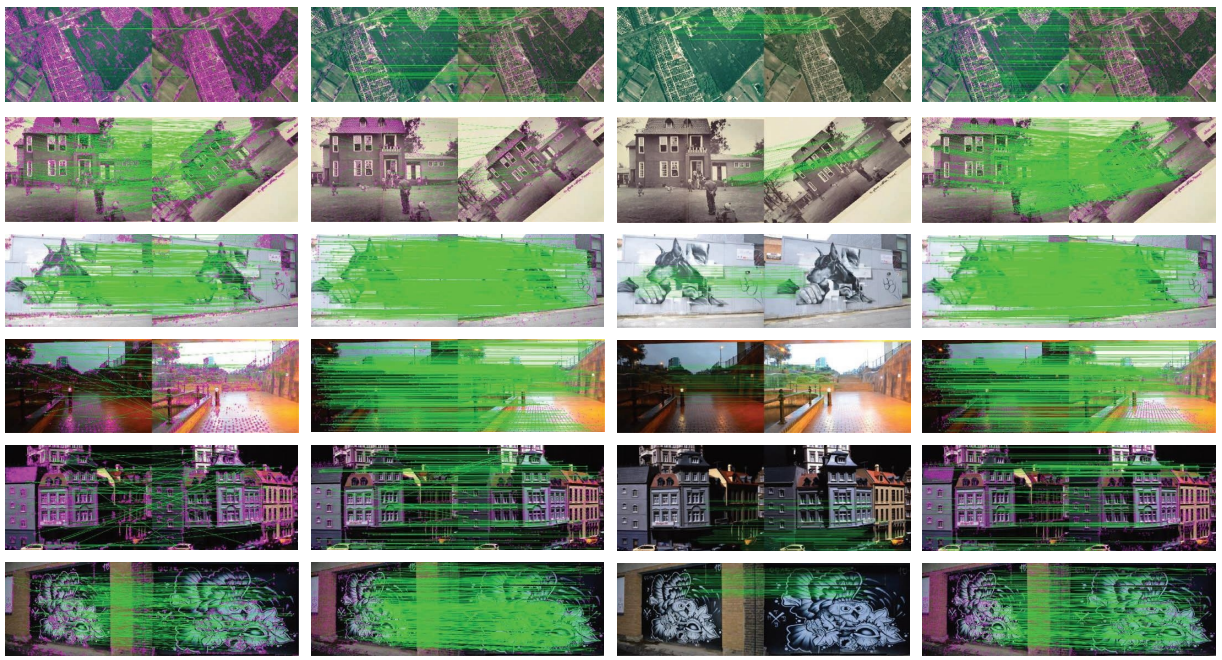


Figure 5: Feature point matching diagram.

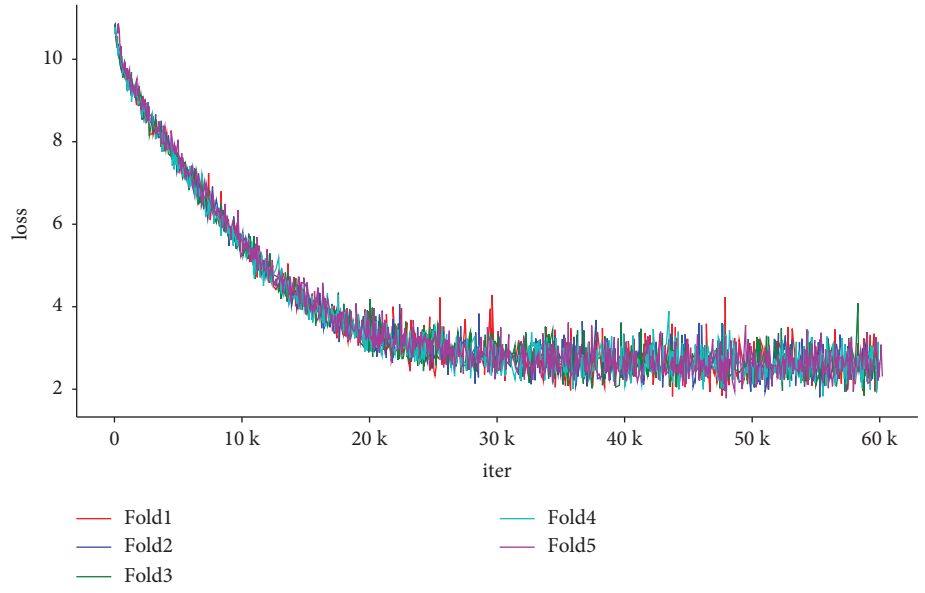
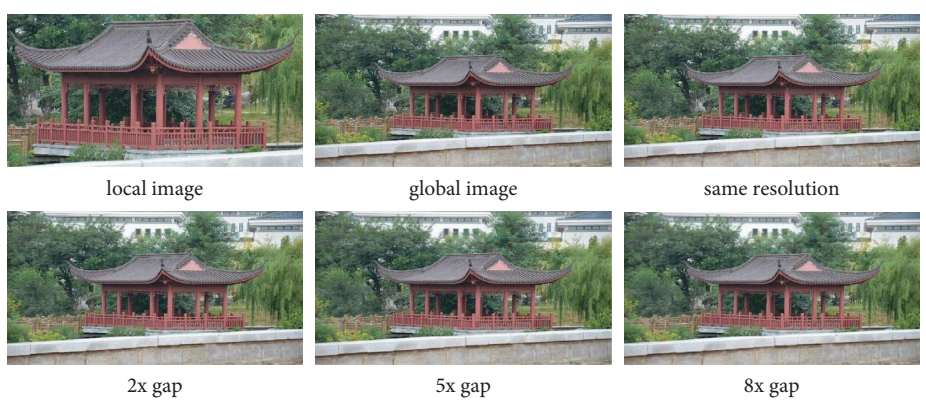


Figure 6: Effectiveness of experiments for random splitting.

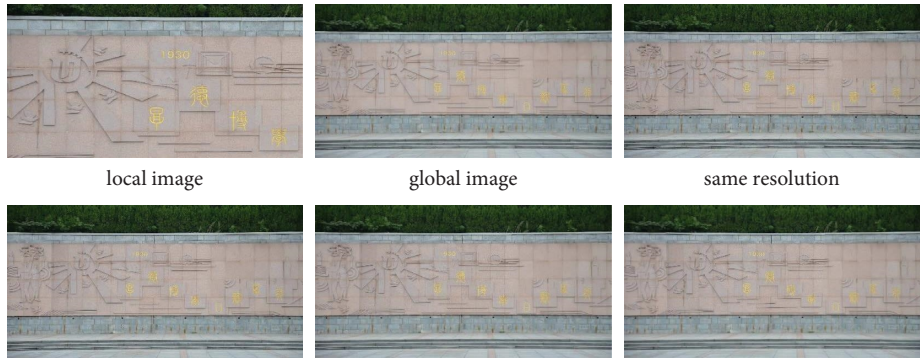
the same scale by scaling. We use the FLANN-based matcher to match the feature points according to the descriptor vector, we use the RANSAC algorithm to filter the false matches in an iterative manner, and we use the projective transformation to obtain the homography transformation matrix. The algorithm uses the transformation matrix to transform the image and finally uses the mask to synthesize the image to realize the image matching of the cross-resolution image. Experiments have shown that our algorithm can achieve a good matching effect whether at the same resolution or at a resolution difference of up to 8 times.

Figure 7: Feature point detection schematic diagram.



(a)

Figure 8: Continued.



(b)

Figure 8: Cross-resolution image matching.

7. Conclusion and Future Work

In this paper, we propose an asymmetric convolution-based feature point detection and description network, which has stronger detection capability for image feature points and descriptors. We propose a new model adaptation technique, which is used together with the homography adaptation technique to generate label datasets with higher accuracy and uses self-supervision to break the reliance on manual labeling. Experiments show that model adaptation techniques are helpful for training network models for sparse and accurate feature point detection and description. However, the ELU activation function that we use can accelerate training during the training phase, but it will increase the weak inferring time. Similarly, the mix pooling layer performs parallel pooling operations, which will also increase the inferring time. Nevertheless, the increased reasoning time does not affect the real-time nature of the model. We also propose a new image-matching method based on the feature points and descriptors detected by ACPoint, which can achieve accurate matching of images across scales.

Same-resolution or cross-resolution image matching tasks help generate high-definition scenes with a large field of view. For higher-resolution image synthesis tasks, how to deal with the color relationship between the pasted image and the background image, as well as the image distortion caused by ultrahigh-resolution lenses, is the focus of our future work.

Data Availability

The HPatches dataset that supports the findings of this study is available at https://icvl.ee.ic.ac.uk/vbalnt/hpatches/. The MSCOCO dataset that supports the findings of this study is available at https://images.cocodataset.org/zips/train2017.zip, https://images.cocodataset.org/zips/val2017.zip, and https://images.cocodataset.org/zips/test2017.zip.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61872170).

References

[1] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[2] Y. Ye, T. Tang, B. Zhu, C. Yang, B. Li, and S. Hao, "A multiscale framework with unsupervised learning for remote sensing image registration," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[3] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image matching from handcrafted to deep features: a survey," *International Journal of Computer Vision*, vol. 129, no. 1, pp. 23–79, 2021.

[4] X. Yuan, L. Fang, Q. Dai, D. J. Brady, and Y. Liu, "Multiscale gigapixel video: a cross resolution image matching and warping approach," in *Proceedings of the 2017 IEEE International Conference on Computational Photography (ICCP) IEEE*, pp. 1–9, Stanford, CA, USA, June 2017.

[5] O. Fried, S. Avidan, and D. Cohen-Or, "Patch2vec: globally consistent image patch representation," in *Computer Graphics Forum*, vol. 36, pp. 183–194, Wiley Online Library, Hoboken, NJ, USA, 2017.

[6] X. Lu, L. Zhu, Z. Cheng, L. Nie, and H. Zhang, "Online multimodal hashing with dynamic query-adaption," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 715–724, Paris France, July 2019.

[7] M. S. Hanif, "Patch match networks: improved two-channel and Siamese networks for image patch matching," *Pattern Recognition Letters*, vol. 120, pp. 54–61, 2019.

[8] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 224–236, Salt Lake City, UT, USA, June 2018.

[9] Q. Chen, J. Xu, and V. Koltun, "Fast image processing with fully-convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2497–2506, Venice, Italy, October 2017.

[10] T. Y. Lin, M. Maire, S. Belongie et al., "Microsoft coco: common objects in context," in *Proceedings of the European*

Conference on Computer Vision, pp. 740–755, Zurich, Switzerland, September 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence*, San Francisco, CA USA, February 2017.

[14] X. Ding, Y. Guo, G. Ding, and J. Han, "Acnet: strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1911–1920, Seoul, Korea, February 2019.

[15] X. Ding, X. Zhang, J. Han, and G. Ding, "Diverse branch block: building a convolution as an inception-like unit," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10886–10895, Montreal, BC, Canada, November 2021.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, Las Vegas, NV, USA, June 2016.

[17] D. I. Barnea and H. F. Silverman, "A class of algorithms for fast digital image registration," *IEEE Transactions on Computers*, vol. C-21, no. 2, pp. 179–186, 1972.

[18] P. Viola and W. M. Wells III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[19] Q. S. Chen, M. Defrise, and F. Deconinck, "Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1156–1168, 1994.

[20] E. De Castro and C. Morandi, "Registration of translated and rotated images using finite Fourier transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 700–703, 1987.

[21] P. Pal, S. Banerjee, A. Ghosh, D. R. Vago, and J. Brewer, "DFT21: discrete fourier transform in the 21st century 2021," 2021, https://www.techrxiv.org/articles/preprint/DFT21_Discrete_Fourier_Transform_in_the_21st_century/16543521.

[22] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 118–126, Santiago, Chile, December 2015.

[23] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, Salt Lake City, UT, USA, December 2018.

[24] L. Nie, C. Lin, K. Liao, M. Liu, and Y. Zhao, "A view-free image stitching network based on global homography," *Journal of Visual Communication and Image Representation*, vol. 73, Article ID 102950, 2020.

[25] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: detector-free local feature matching with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8922–8931, Nashville, TN, USA, November 2021.

[26] C. Liu, J. Yuen, and A. J. Torralba, "SIFT flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, 2011.

[27] C. B. Choy, J. Y. Gwak, S. Savarese, and M. Chandraker, "Universal correspondence network," 2016, https://arxiv.org/abs/1606.03558.

[28] I. Rocco, M. Cimpoi, R. Arandjelovi, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," 2018, https://arxiv.org/abs/1810.10510.

[29] P. E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4938–4947, Seattle, WA, USA, August 2020.

[30] P. C. Ng and S. Henikoff, "SIFT: predicting amino acid changes that affect protein function," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3812–3814, 2003.

[31] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[32] E. Karami, S. Prasad, and M. Shehata, "Image matching using sift, surf, brief and orb: performance comparison for distorted images," 2017, https://arxiv.org/abs/1710.02726.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[34] D. R. Roosab, E. H. Shiguemori, and A. C. Lorenac, *Comparing ORB and AKAZE for Visual Odometry of Unmanned Aerial Vehicles*, Semantic Scholar, Seattle, WA, USA, 2016.

[35] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the European Conference on Computer Vision*, pp. 430–443, Graz, Austria, May 2006.

[36] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: learned invariant feature transform," in *Proceedings of the European Conference on Computer Vision*, pp. 467–483, Amsterdam, Netherlands, October 2016.

[37] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1538–1546, Boston, MA, USA, October 2015.

[38] Z. Yang, T. Dan, and Y. Yang, "Multi-temporal remote sensing image registration using deep convolutional features," *IEEE Access*, vol. 6, pp. 38544–38555, 2018.

[39] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, Las Vegas, NV, USA, June 2016.

[40] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, *TILDE: a temporally invariant learned detector 2015*, in *Proceedings of the [IEEE 2015 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5279–5288, Boston, MA, USA, June 2015.

[41] V. Balntas, K. Lenc, A. Vedaldi, and 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*, pp. 5173–5182, Honolulu, HI, USA, July 2017.

[42] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning PMLR*, pp. 448–456, Lille, France, July 2015.

[43] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "ast and accurate deep network learning by exponential linear units (elus)," 2015, https://arxiv.org/abs/1511.07289.

[44] Y. Y. Jau, R. Zhu, H. Su, and M. Chandraker, "Deep keypoint-based camera pose estimation with geometric constraints," in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) IEEE*, pp. 4950–4957, Las Vegas, NV, USA, October 2020.