SuperPoint: Self-Supervised Interest Point Detection and Description

Author	Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich
PDF	Self-Supervised Interest Point Detection and Description.pdf
Score	
Completed	
Status	In progress
Type	Computer Vision Deep Learning

3. SuperPoint Architecture

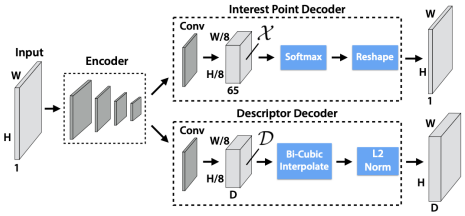


Figure 3. **SuperPoint Decoders**. Both decoders operate on a shared and spatially reduced representation of the input. To keep the model fast and easy to train, both decoders use non-learned upsampling to bring the representation back to $\mathbb{R}^{H \times W}$.

Innovation

Most of the network's parameters are shared between the two tasks, which is a departure from traditional systems which first detect interest points, then compute descriptors and lack the ability to share computation and representation across the two tasks.

DataSet

- MS-COCO

MS COCO (Microsoft Common Objects in Context) is a large-scale image recognition and scene understanding dataset for computer vision. It was created by Microsoft Research and contains over 330,000 images with 2.5 million object instances labeled with 80 object categories, including common objects such as people, animals, furniture, and vehicles.

3. SuperPoint Architecture

3.1. Shared Encoder(VGG-style)

- Consist with convolutional layer and spatial downsampling via pooling and non-linear activation functions
  - Use convolution layers to extract the feature of the image
    - Convolution size: $H_c = H/8,\ W_c = W/8$ for image size $W \cdot H$
    - Produce 8×8 cells
  - Use 2×2 non-overlapping max pooling operation in 64 cells
    - Function: extract the edges
- Function: Use to reduce the demensionality of the image

Input$(R^{W \times H})$ → Output$(R^{W_C \times H_C \times 64})$

- The encoder maps the input image $I \in R^{H \times W}$ to an intermediate tensor $B \in R^{H_c \times W_c \times 65} B \in R^{H_c \times W_c \times F}$ with smaller spatial dimension and greater channel depth $(i.e., H_c < H, W_c < W \ and \ F > 1).$

3.2 Interest Point Decoder

Inputs $(R^{W_C \times H_C \times 65})$ → Output$(R^{W \times H})$

- 65 chanel correspond to
  - local non-overlapping 8*8 grid regions of pixels
  - extra “no interest point” dustbin
- Then apply a chanel-wise softmax activation function
- The dusbin chanel is removed$(R^{W_C \times H_C \times 65}) \rightarrow (R^{W \times H} \times 64)$
- Dusbin
  - In SuperPoint, the "dusbin" refers to the dual SuperPoint binarization. It is a technique used in the algorithm to improve the robustness of the keypoint extraction process.
  - In computer vision, the keypoint extraction process is used to identify and describe distinctive points in an image that are invariant to geometric and photometric transformations. To achieve this, the SuperPoint algorithm uses a combination of a feature extractor network and a descriptor network, both trained end-to-end on a large dataset of images.
  - The dual SuperPoint binarization is used to improve the robustness of the feature extractor network by creating two separate binary maps from the activations of the network. These binary maps are then combined to form a final binary map that is used to identify the keypoints in the image. The use of the dual SuperPoint binarization allows the algorithm to better handle changes in viewpoint, illumination, and partial occlusion, making it more robust and effective in a variety of real-world scenarios.

- $R^{W_C \times H_C \times 64} \rightarrow R^{W \times H}$
  - each pixel of the output corresponds to a probability of “point-ness” for that pixel in the input. Once the probability exceed a threshold, the pixiel will be detected as a interest point.

3.3 Descriptor Decoder

- The descripor head will output a Tensor which has $R^{\mathcal{H}_c \times \mathcal{W}_c \times \mathcal{D}}$. In this paper, the author sets $\mathcal{D} = 256$.
- Bicubic interpolation

最近邻插值、双线性插值与双三次插值

最近在看一些有关超分辨率的文章，其中都有提到resize的操作。之前一直是直接调用python中的imutils库来完成，算法的一些细节并不是很清楚，今天在网上查了一下，算是搞懂个大概，也就把它写下来了。 文章总共分为三个部分，见下面目录。第一次写知乎，还不太熟练。 ...

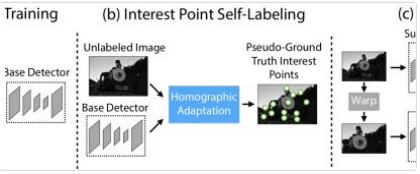
知 https://zhuanlan.zhihu.com/p/428523385

3.4 Loss Function

Notes on SuperPoint

SuperPoint: Self-Supervised Interest Point Detection and Description 这是我看的第一篇特征点检测的文章... 这篇文章是 Magic Leap 的。 文章的主要贡献给出了一种不需要 human annotation，以 Self-Supervised 训练的 fully-convolutional CNN 来同时学 interest point detectors and descriptors。 Self-Supervised 是以 From Simple

知 https://zhuanlan.zhihu.com/p/54969632



- it is consisted with 2 part:
  - Formula

$$\mathcal{L}_(\mathcal{X},\mathcal{X}',\mathcal{D},\mathcal{D}';\mathcal{Y},\mathcal{Y}',\mathcal{S}) = \mathcal{L}_p(\mathcal{X},\mathcal{Y}) + \mathcal{L}_p(\mathcal{X}',\mathcal{Y}') + \lambda\mathcal{L}_d(\mathcal{D},\mathcal{D}',\mathcal{S})$$

  - λ is used to balance the final loss
- Interest point detector loss function
  - Cross-entropy loss:

Cross-Entropy Loss Function

Consider a 4-class classification task where an image is classified as either a dog, cat, horse or cheetah. In the above Figure, Softmax converts logits into probabilities. The purpose of the Cross-Entropy is to take the output probabilities (P) and measure the distance from the truth values (as shown in Figure below).

tds https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

  - Formula
  - where t_i is the truth label and p_i is the softmax probability for i^{th} class

$$H(P,Q) = -\sum_{x=1} t(x) log\ p(x)$$

  - For example
  - expected output: [1,0], model output: [0.7, 0.3]

$$H(P,Q) = -(1 \times log0.7) - (0 \times log0.3) = -log0.7 = 0.15490195998$$

  - In SuperPoint model
    - Softmax and cross entropy in a cell $h * w$

$$l_p(\mathbf{x}_{hw}; y) = -\log\left(\frac{\exp(\mathbf{x}_{hwy})}{\sum_{k=1}^{65}\exp(\mathbf{x}_{hwk})}\right). \qquad (3)$$

  - Descriptor loss

- The descriptor loss is applied to all pairs]of descriptor cells, $d_{hw} \in D$ from the first image and $d'_{h'w'} \in$ D0 from the second image.
- The contrastive loss can be expressed mathematically as:

"

- GPT explaination

The loss function in SuperPoint is a measure of the difference between the predicted keypoints and descriptors and the ground-truth annotations. The loss function is used to guide the training process of the feature extractor and descriptor networks, so that the network parameters are optimized to minimize the difference between the predicted and ground-truth keypoints and descriptors.

There are two main components to the loss function in SuperPoint: the keypoint loss and the descriptor loss.

1. Keypoint loss: The cross-entropy loss is a commonly used loss function in supervised learning problems, where the goal is to classify the input data into one of several categories. In the case of SuperPoint, the keypoint loss is used to classify each pixel in the image as either a keypoint or a non-keypoint.

   $L_{keypoint} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$

   where N is the number of pixels in the image, y_i is the ground-truth label for pixel i (1 for keypoint, 0 for non-keypoint), and p_i is the predicted probability that pixel i is a keypoint.

   where N is the number of keypoints, p_i is the ground-truth keypoint position, and p_i' is the predicted keypoint position. The keypoint loss is typically a mean squared error (MSE) loss, which measures the average squared difference between the predicted and ground-truth keypoints.

2. Descriptor loss: The descriptor loss is a measure of the difference between the predicted descriptors and the ground-truth descriptors. The descriptor loss can be expressed mathematically as:

   $L_{descriptor} = \frac{1}{N} * sum_{i=1}^{N} ||d_i - d_i'||^2$,

   where N is the number of keypoints, d_i is the ground-truth descriptor, and d_i' is the predicted descriptor. The descriptor loss is also typically a MSE loss, which measures the average squared difference between the predicted and ground-truth descriptors.

The total loss function in SuperPoint is a combination of the keypoint loss and the descriptor loss:

$$L = L_{keypoint} + L_{descriptor}$$

The network parameters are optimized to minimize the total loss function during the training process, so that the predicted keypoints and descriptors are as close as possible to the ground-truth annotations. The optimization process can be performed using gradient descent or another optimization algorithm, such as Adam or SGD.

- mAP(mean of Average precision)

Mean Average Precision (mAP) is a common evaluation metric used in object detection and image retrieval tasks in computer vision. mAP measures the average precision of a model over a set of images, where precision is defined as the ratio of true positive detections to the total number of positive detections (true positive and false positive).

The formula for mAP is defined as:

$$mAP = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{|D_q|} \sum_{k=1}^{|D_q|} \text{Precision}(k) \cdot [\text{Rel}(k) = 1]$$

where Q is the number of query images, D_q is the set of ground-truth bounding boxes for query image q, $|D_q|$ is the number of ground-truth bounding boxes in D_q, $\text{Rel}(k)$ is a binary variable indicating whether the k-th prediction is a true positive (1) or not (0), and $\text{Precision}(k)$ is the precision of the first k predictions.

The precision of the first k predictions is defined as the ratio of the number of true positive detections to the total number of positive detections (true positive and false positive) up to the k-th prediction:

$\text{Precision}(k) = \frac{\sum_{i=1}^{k}[\text{Rel}(i)=1]}{k}$

mAP provides a single scalar value that summarizes the performance of a model on a set of images, and it is widely used in computer vision benchmarks, such as the PASCAL VOC and MS COCO datasets, to compare the performance of different models. A higher mAP value indicates a higher level of accuracy and recall in object detection and image retrieval tasks.

- Repeatability

Repeatability is typically expressed as a percentage of correctly detected and described keypoints. Given a set of images of the same scene, the repeatability R can be calculated as:

$R = \frac{\sum_{i=1}^{n} N_{correct}^{i}}{\sum_{i=1}^{n} N_{total}^{i}} \times 100$

where n is the number of images in the set, $N_{correct}^{i}$ is the number of correctly detected and described keypoints in image i, and N_{total}^{i} is the total number of keypoints detected in image i.

The calculation of repeatability involves the comparison of keypoints and descriptors across multiple images of the same scene, and it typically uses a distance metric, such as the Euclidean distance, to compare the descriptors of keypoints in different images. A threshold value is used to determine whether a keypoint is considered correctly detected and described, based on the distance between the descriptors of the keypoint in different images.

NMS(Non-Maximum Suppression)

Non-Maximum Suppression (NMS) is a technique used in computer vision to reduce the number of detections of an object in an image, while preserving the highest-scoring detections. NMS is typically used in object detection algorithms to remove overlapping bounding boxes that correspond to the same object, so that each object is represented by a single detection.

The basic idea behind NMS is to keep the highest-scoring detection and remove all other detections that have a high overlap with the highest-scoring detection. The overlap between two bounding boxes is typically measured using the Intersection over Union (IoU) metric, which is defined as:

$\text{IoU}(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$

where A and B are two bounding boxes, and $\text{area}(\cdot)$ is the area of a bounding box. The IoU metric measures the overlap between two bounding boxes as the ratio of their intersecting area to their union area.

NMS works by first selecting the highest-scoring detection, and then removing all other detections that have an IoU with the highest-scoring detection greater than a threshold value t. The process is repeated until all detections have been processed.

NMS is an important technique in object detection algorithms because it helps to reduce the number of false positive detections and to improve the accuracy of the detections. By removing overlapping detections, NMS helps to ensure that each object is represented by a single, high-scoring detection, which is important for tasks such as object tracking and recognition.

5. Homographic adaptation

```
def one_adaptation(net, raw_image, probs, counts, images, config, device='cpu'):
    """
    :param probs:[B,1,H,W]
    :param counts: [B,1,H,W]
    :param images: [B,1,H,W,N]
    :return:
    """
    """
    B: Batch size, which is the number of samples in a batch of data.

    C: Number of channels, which is the number of features or channels in an image. For example, in a grayscale image, C would be 1, and in

    H: Height, which is the number of rows in an image.

    W: Width, which is the number of columns in an image.
    """
    B, C, H, W, _ = images.shape
    #sample image patch
```

```
    M = sample_homography(shape=[H, W], config=config['homographies'],device=device)
    M_inv = torch.inverse(M)
    ##
    warped = kornia.warp_perspective(raw_image, M, dsize=(H,W), align_corners=True)
    mask = kornia.warp_perspective(torch.ones([B,1,H,W], device=device), M, dsize=(H, W), mode='nearest',align_corners=True) # H
    count = kornia.warp_perspective(torch.ones([B,1,H,W],device=device), M_inv, dsize=(H,W), mode='nearest',align_corners=True) # H^{-1}

    # Ignore the detections too close to the border to avoid artifacts
    if config['valid_border_margin']:
        ##TODO: validation & debug
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (config['valid_border_margin'] * 2,) * 2)
        kernel = torch.as_tensor(kernel[np.newaxis,:,:], device=device)#BHW
        kernel = torch.flip(kernel, dims=[1,2])
        _, kH, kW = kernel.shape
        origin = ((kH-1)//2, (kW-1)//2)
        count = erosion2d(count, kernel, origin=origin) + 1.
        mask = erosion2d(mask, kernel, origin=origin) + 1.
    # filter used to remove the interest point close to margin
    mask = mask.squeeze(dim=1)#B,H,W
    count = count.squeeze(dim=1)#B,H,W


    # Predict detection probabilities
    prob = net(warped)
    prob = prob['prob']
    prob = prob * mask # f(H(image))
    prob_proj = kornia.warp_perspective(prob.unsqueeze(dim=1), M_inv, dsize=(H,W), align_corners=True)
    prob_proj = prob_proj.squeeze(dim=1)#B,H,W
    prob_proj = prob_proj * count#project back: H^{-1} * f(H(image))
    ##

    probs = torch.cat([probs, prob_proj.unsqueeze(dim=1)], dim=1)#the probabilities of each pixels on raw image
    counts = torch.cat([counts, count.unsqueeze(dim=1)], dim=1)
    images = torch.cat([images, warped.unsqueeze(dim=-1)], dim=-1)

    return probs, counts, images
```

This code is a function that performs homography adaptation on an image. The function takes as input a neural network `net`, the raw image `raw_image`, and several other tensors `probs`, `counts`, and `images`, as well as a configuration dictionary `config` and a device for running the computations (either the CPU or GPU).

The function first samples a homography matrix `M` using the `sample_homography` function. The homography matrix is used to warp the raw image `raw_image` using the `kornia.warp_perspective` function from the Kornia library. The warped image is then passed through the neural network `net` to predict the detection probabilities.

The function also computes a binary mask tensor `mask` and a count tensor `count` that keep track of the number of valid detections in the image. The `mask` and `count` tensors are warped using the homography matrix and its inverse, respectively.

The code then checks the `valid_border_margin` setting in the `config` dictionary. If the `valid_border_margin` is set, the code performs morphological erosion on the `count` and `mask` tensors using the `erosion2d` function. The erosion is performed using an elliptic structuring element, as specified by the `cv2.MORPH_ELLIPSE` argument to `cv2.getStructuringElement`. The purpose of the erosion is to remove pixels that are too close to the border of the image, as specified by the `valid_border_margin` setting.

Finally, the function concatenates the predicted probabilities `prob_proj`, the count tensor `count`, and the warped image `warped` with the corresponding tensors `probs`, `counts`, and `images`, respectively. The function returns the updated tensors `probs`, `counts`, and `images`.

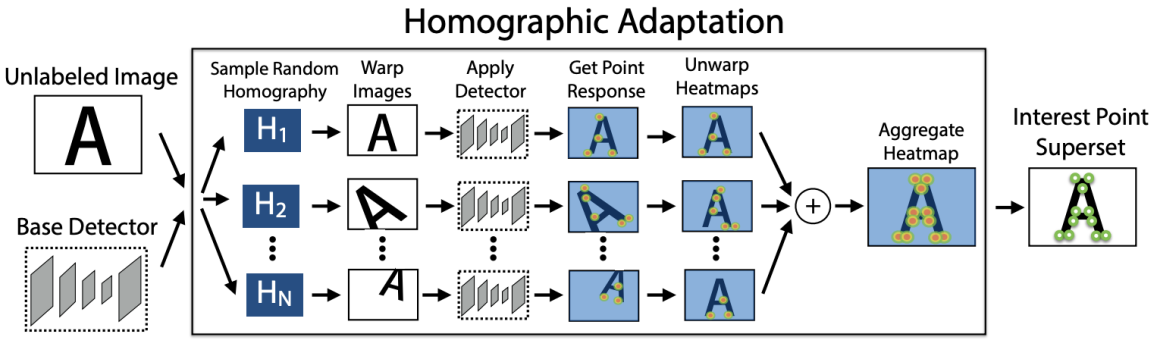


Figure 5. **Homographic Adaptation.** Homographic Adaptation is a form of self-supervision for boosting the geometric consistency of an interest point detector trained with convolutional neural networks. The entire procedure is mathematically defined in Equation 10.

Homographic adaptation is a technique used in the SuperPoint algorithm to align keypoints across multiple images taken from different viewpoints. The technique uses a homography matrix to map one image to the other, so that corresponding keypoints in the two images are aligned.

In SuperPoint, the homography matrix is estimated from a set of corresponding keypoints, and it is used to align the keypoints in the two images. The aligned keypoints are then used to compare the descriptors of the keypoints, and to determine whether they correspond to the same feature in the two images.

The homography matrix can be estimated using a variety of methods, including the RANSAC (Random Sample Consensus) algorithm, which is a robust estimation technique that can handle outliers and noisy data. The RANSAC algorithm randomly selects a set of corresponding keypoints, and it estimates the homography matrix that best aligns the keypoints. The algorithm is repeated multiple times, and the homography matrix that results in the highest number of inliers (correctly aligned keypoints) is selected.

Once the homography matrix has been estimated, it can be used to align the keypoints in the two images, by transforming the keypoints in one image using the homography matrix. The aligned keypoints can then be used to compare the descriptors of the keypoints, and to determine whether they correspond to the same feature in the two images.

Homographic adaptation is an important step in the SuperPoint algorithm, as it allows the algorithm to align keypoints across multiple images taken from different viewpoints, which is important for tasks such as object detection, recognition, and structure-from-motion. The technique is robust and can handle a range of geometric and photometric transformations, such as rotations, scalings, and changes in illumination and exposure.

Average response

The average response of a feature detector or keypoint descriptor is calculated as the mean value of the response or activation values over a set of feature detectors or keypoint descriptors.

Mathematically, the average response R_{avg} can be expressed as:

$$R_{avg} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

where N is the number of feature detectors or keypoint descriptors in the set, and R_i is the response or activation value of the i^{th} feature detector or keypoint descriptor.

The average response is a simple and effective metric for evaluating the performance of feature detection and description algorithms, and it provides a measure of the overall quality of the features and descriptors. A higher average response indicates a higher quality of the features and descriptors, and it is an important factor to consider in the selection and comparison of feature detection and description algorithms for computer vision applications.

Morphology Erosion

（三十二）形态学----膨胀和腐蚀

时间为友，记录点滴。 我们在《 初始滤波之均值滤波》中有聊过滤波的本质，以及介绍了其中一种线性滤波（均值滤波）。对于常见的非线性滤波"中值滤波"也在《视频的读取和处理》中有介绍。今天我们来介绍另外两种…

https://zhuanlan.zhihu.com/p/83078037

- Definition of structure element

Enumeration Type Documentation

◆ MorphShapes

enum cv::MorphShapes

`#include <opencv2/imgproc.hpp>`

shape of the structuring element

Enumerator	
MORPH_RECT Python: cv.MORPH_RECT	a rectangular structuring element: $E_{ij} = 1$
MORPH_CROSS Python: cv.MORPH_CROSS	a cross-shaped structuring element: $E_{ij} = \begin{cases} 1 & \text{if } i = \text{anchor.y or } j = \text{anchor.x} \\ 0 & \text{otherwise} \end{cases}$
MORPH_ELLIPSE Python: cv.MORPH_ELLIPSE	an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle Rect(0, 0, esize.width, 0.esize.height)