



Image Matching Challenge

2022 Kaggle竞赛复盘

Sydney huo

@七月在线116组

总目录

讲义1

- 01 理解比赛
- 02 理解kaggle

讲义2

- 01 Find Good Correspondences
- 02 初识 Kornia

讲义3

- 01 SuperGlue vs LoFTR
- 02 数据增强

讲义4

- 01 MAGSAC++
- 02 Matchfomer
- 03 模型融合



讲义1

- 01 理解比赛
- 02 理解Kaggle

[Description](#)[Evaluation](#)[Timeline](#)[Prizes](#)[Code Requirements](#)[CVPR 2022 Workshop](#)[Problem Definition](#)

For most of us, our best camera is part of the phone in our pocket. We may take a snap of a landmark, like the Trevi Fountain in Rome, and share it with friends. By itself, that photo is two-dimensional and only includes the perspective of our shooting location. Of course, a lot of people have taken photos of that fountain. Together, we may be able to create a more complete, three-dimensional view. What if machine learning could help better capture the richness of the world using the vast amounts of unstructured image collections freely available on the internet?

The process to reconstruct 3D objects and buildings from images is called Structure-from-Motion (SfM). Typically, these images are captured by skilled operators under controlled conditions, ensuring homogeneous, high-quality data. It is much more difficult to build 3D models from assorted images, given a wide variety of viewpoints, lighting and weather conditions, occlusions from people and vehicles, and even user-applied filters.



01

理解比赛 | 比赛背景

Google在谷歌地图中采用了Structure-from-Motion技术，例如从街景和航拍图像创建的3D模型。为了加速对该主题的研究，并更好地利用已经公开的数据量，谷歌与不列颠哥伦比亚大学和捷克技术大学合作举办了这场比赛。

从图像重建 3D 对象和建筑物的过程称为Structure-from-Motion (SfM)。通常，这些图像是由熟练的操作员在受控条件下捕获的，以确保数据的均匀、高质量。考虑到各种各样的视点、照明和天气条件、人和车辆的遮挡，甚至是用户应用的过滤器，从各种图像构建 3D 模型要困难得多。

理解比赛 | 比赛背景

比赛的问题

问题的第一部分是识别两个图像捕获场景的相同物理点，例如窗口的角点。可以通过局部特征（图像中可以在不同视图中可靠识别的关键位置）来实现。局部特征包含捕获兴趣点周围外观的简短描述向量。通过比较这些描述符，可以在两个或多个图像的图像位置的像素坐标之间建立可能的对应关系。这种“图像配准”使得通过三角测量恢复点的 3D 位置。

你将帮助解决计算机视觉中的这个众所周知的问题，令使用非结构化图像集合绘制世界地图成为可能。



Problem definition

The goal of the contest is to estimate the relative pose between two images. This requires some knowledge of projective and epipolar geometry, particularly with regards to the following:

The calibration matrix captures camera properties that determine the transformation between 3D points and 2D (pixel) coordinates. It is also known as the camera intrinsics.

The rotation matrix and the translation vector capture the 6-degree-of-freedom pose (position and orientation) of the camera in a global reference frame. They are collectively known as the camera extrinsics.

The fundamental matrix encapsulates the projective geometry between two views of the same scene. It is not affected by the contents in the scene, and depends only on the intrinsics and extrinsics of the two cameras.

The training data provides , and as ground truth. Participants are asked to estimate . We explain these concepts in more detail below.

Projective geometry

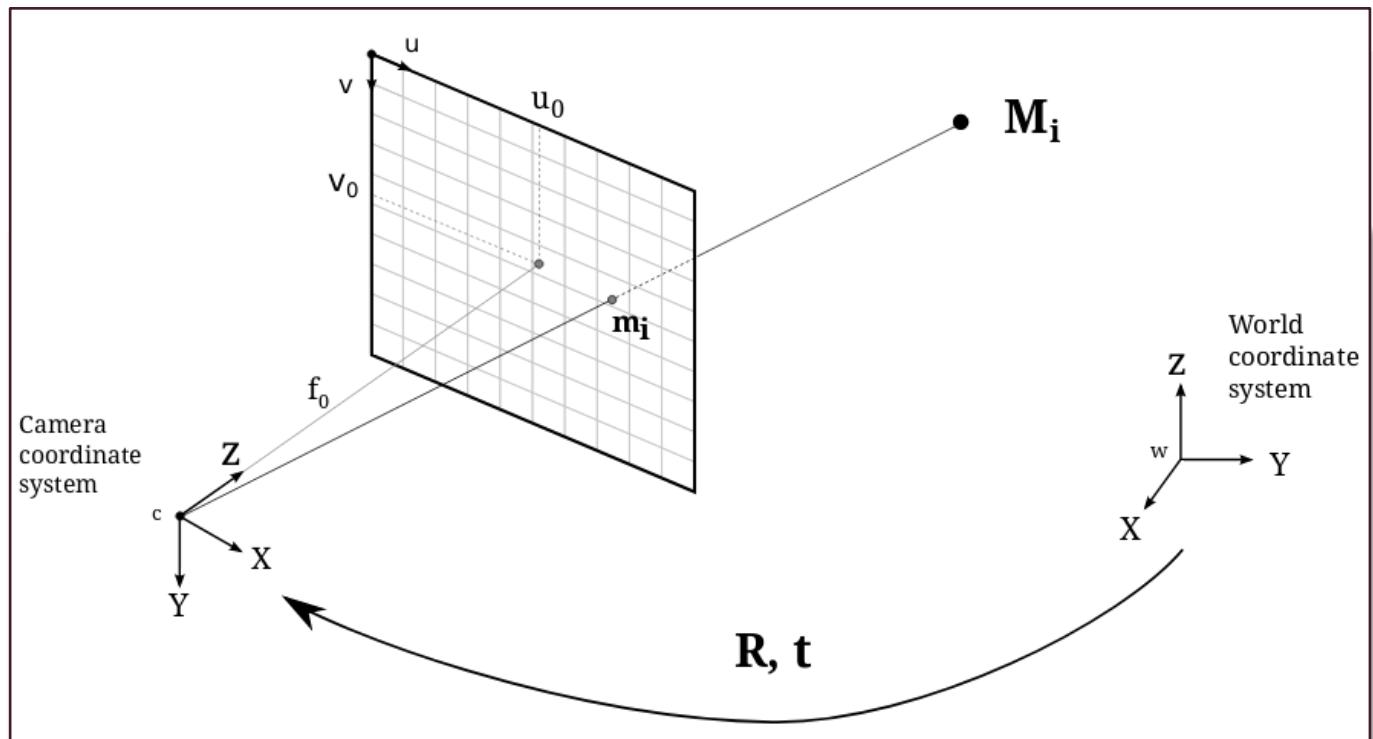
In computer vision, the transformation between 3D (world) and 2D (image) is governed by projective geometry. We use the simplest camera model, that of the pinhole camera, which is illustrated below.

比赛背景

比赛目标

比赛的目标是估计两个图像之间的相对姿势。这需要一些投影和对极几何知识，特别是在以下方面：

- 校准矩阵 \mathbf{K} 捕获确定 3D 点和 2D（像素）坐标之间转换的相机属性。它也被称为相机内在函数。
- 旋转矩阵 \mathbf{R} 和翻译向量在全局参考框架中捕获相机的 6 自由度姿态（位置和方向）。它们统称为相机外在。
- 基本矩阵 \mathbf{F} 封装了同一场景的两个视图之间的投影几何。它不受场景中内容的影响，只取决于两个相机的内在和外在。

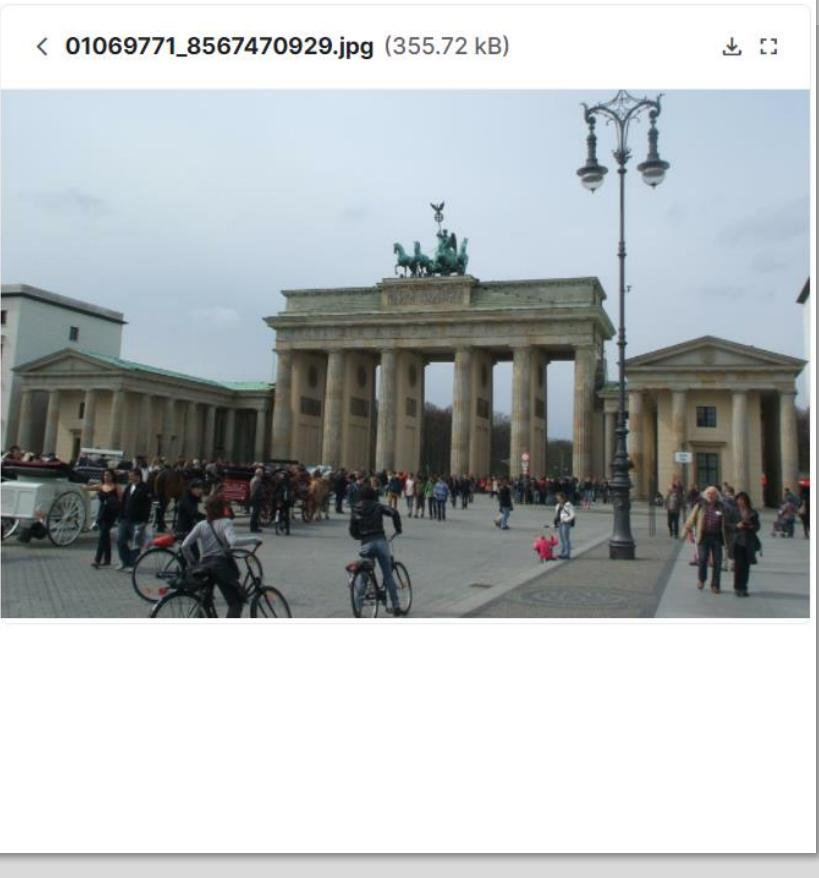


比赛数据

Data Explorer
2.65 GB

- test_images
- train
 - brandenburg_gate
 - images
 - 00883281_963...
 - 01069771_8567...
 - 02035158_8518...
 - 02749394_386...
 - 02936509_948...
 - 03058406_697...
 - 03300929_279...
 - 03774918_7196...
 - 03890690_272...
 - 04683648_247...
 - 04684290_269...
 - 04692004_396...
 - 04705241_3760...
 - 04803880_1111...
 - 05712502_4051...
 - 05714706_2362...
 - 05767185_3704...
 - 06809631_549...

Summary
5720 files
120 columns



本次比赛数据规模如下：

训练集： 5678多张图片

测试集： 隐藏的测试集中大约 10,000 对图像

预测的形式是一个fundamental matrix，封装了同一场景的两个视图之间的投影几何

测试数据与训练数据来源不同，并且包含大部分城市市场的照片，具有不同程度的重叠。形成一对的两个图像可能相隔数月或数年收集，但绝不会少于 24 小时。弥合这一领域差距是竞争的一部分。图像已调整大小，最长边缘约为 800 像素，可能具有不同的纵横比（包括纵向和横向），并且是直立的。

比赛数据

train/*/calibration.csv

- `image_id` – 图像文件名。
- `camera_intrinsics` – 图像的3*3校准矩阵K。通过行索引将其展平为向量。
- `rotation_matrix` – 图像的3*3旋转矩阵R。通过行索引将其展平为向量。
- `translation_vector` – 平移向量T。

train/*/pair_covisibility.csv

- `pair` – 标识一对图像的字符串，编码为两个图像文件名（不带扩展名），由连字符分隔，如key1–key2，其中key1 > key2。
- `covisibility`: 估计两个图像之间的重叠。数字越大表示重叠越大。我们建议使用所有具有 0.1 或更高共可见性估计值的对。
- `fundamental_matrix`: 来自校准文件的目标列。

比赛数据

train/scaling_factors.csv

- 通过Structure-from-Motion重建的每个场景的姿势，并且只能精确到比例因子。该文件包含每个场景的标量，可用于将它们转换为距离。

train/*/images

- 在同一位置附近拍摄的一批图像。

train/LICENSE.txt

- 每个图像的具体来源和许可记录。

test.csv

- sample_id – 图像对的唯一标识符。
- batch_id – 批次编号。
- image_[1/2]_id: 对中每个图像的文件名。

评价标准

本次竞赛采用了 mAA

Accuracy = (预测正确的样本数)/(总样本数)=
 $(TP+TN) / (TP+TN+FP+FN)$

本次比赛中：

由于数据集包含多个场景，可能有不同数量的对， 我们为每个场景分别计算该指标， 然后对其进行平均。

我们根据旋转计算误差 (R, 度数) 和平移 (T, 以米为单位)。给定每个阈值，如果姿势满足两个阈值，我们将其分类为准确。我们在十对阈值上执行此操作，一次一对。例如，在 1度和 20 厘米在最好的水平，和 10度和 5 m 在最粗的水平

mAA (mean Average Accuracy)

通过设定不同的阈值，计算达到阈值的样本的Accuracy，并对所有阈值的结果进行平均。

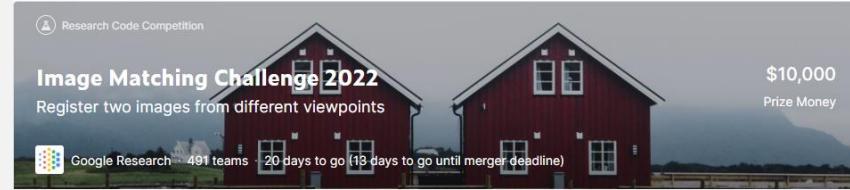
```
def ComputeMaa(err_q, err_t, thresholds_q, thresholds_t):  
    """Compute the mean Average Accuracy at different thresholds, for one scene."""  
  
    assert len(err_q) == len(err_t)  
  
    acc, acc_q, acc_t = [], [], []  
    for th_q, th_t in zip(thresholds_q, thresholds_t):  
        acc += [(np.bitwise_and(np.array(err_q) < th_q, np.array(err_t) < th_t)).sum() /  
len(err_q)]  
        acc_q += [(np.array(err_q) < th_q).sum() / len(err_q)]  
        acc_t += [(np.array(err_t) < th_t).sum() / len(err_t)]  
    return np.mean(acc), np.array(acc), np.array(acc_q), np.array(acc_t)
```

讲义2

- 01 Find Good Correspondences
- 02 初识 Kornia



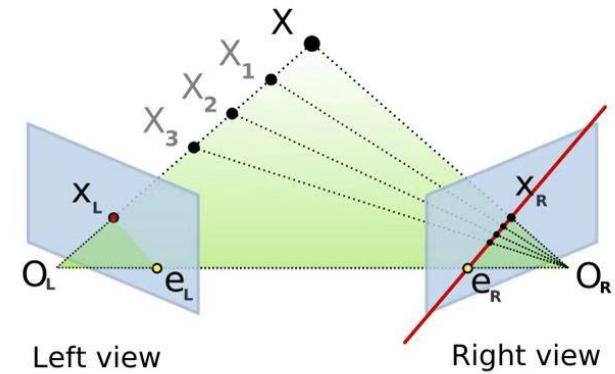
讲义2 | Find Good Correspondences



3D computer vision

- 立体几何是3D computer vision的基础，这里我们简要介绍下对极几何；
 - 对极几何
- 如果仅看一个相机，我们并不能知道深度信息，可如果有两个相机的话（就像人有两只眼睛）我们就能得到深度的信息，如右图所示 O_L 和 O_R 是两个相机中心， X 是物体所在，如果我们只看左边图像上的点 X_L ，我们不能知道物体到底是在哪，点 X_1 、 X_2 或其他地方，可有了右边图像上的 X_R 我们就能得到物体点 X 。
- 在右图，我们把两相机中心的 O_L 和 O_R 连线成为基线，把他们与观测物体的平面 X O_L O_R 成为对极平面，对极平面与两相机图像的交线 $|l$ 和 $|l'$ 称为对极线，而 O_L 和 O_R 与两图像的交点 e_L , e_R 就是对极点。
- 随着观测点 X 的上下移动，对极平面也会围绕基线旋转，而对极平面旋转时对极点是不变的，而在相机图像上所有对极线都会交于对极点，这个对极点就是另一个相机中心在其图像上的像，同样的对极点可以在图像外。

The Epipolar Geometry



Given X_L, X_R must lie on the epipolar line determined by O_L, O_R, X, n_L , and n_R

The Epipolar constraints represented by the Fundamental Matrix between two cameras

对极几何中最重要的一个公式是对极约束 (Epipolar Constraint)，下面让我们来推导出这个约束条件。

根据针孔相机模型，相机成像平面一点的像素坐标 p 和该点在世界坐标系下的3D坐标 P 有 $p = KP$ 的关系，如果用齐次坐标表示则有：

$$dp = KP$$

其中 d 是空间点深度（为了将 p 的齐次项变为1）， K 是相机内参数矩阵， p 和 P 都是齐次坐标。

于是如果以第一个相机的坐标系为参照，对于两个相机则有：

$$d_0p_0 = KP, d_1p_1 = K(RP + t)$$

其中 R 为旋转矩阵 (Rotation)， t 为平移向量 (Translation)。令 $x = K^{-1}p$ ，去掉内参 K 归一化成：

$$d_0x_0 = P, d_1x_1 = RP + t$$

由这两式得：

$$d_1x_1 = R(d_0x_0) + t$$

两边同时乘 t 消去加号后面单独的 t 项：

$$t \times d_1x_1 = t \times Rd_0x_0 + t \times t$$

进而：

$$t \times d_1x_1 = t \times Rd_0x_0$$

再在两边同时左乘一个 x_1 ：

$$x_1^T(t \times d_1x_1) = x_1^Tt \times Rd_0x_0$$

由于等号左边 $\times 1$ 乘上了一个和自身垂直的向量，所以等于0，故：

$$x_1^Tt \times Rx_0 = 0$$

该等式称为对极约束 (Epipolar Constraint)。

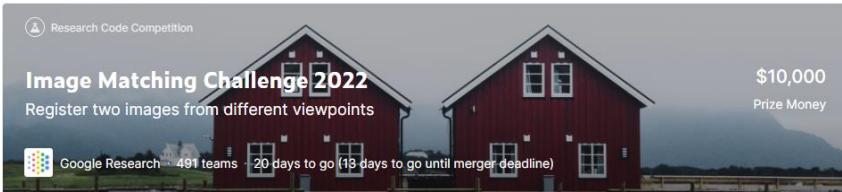
对极约束的几何意义： x_1, t, Rx_0 三者混合积为0，表示这三个向量共面，即上图中三角形的三边共面。

令 $E = t \times R$ ，得到对极约束的新形式：

$$x_1^TEx_0 = 0$$

E 称为本质矩阵 (Essential Matrix)，由外参数 R 和 t 决定。

本质矩阵的几何意义： $x_1^Tl_1 = 0$ ，即 x_1 在直线 $l_1 = Ex_0$ 上，表示 E 将 x_0 投影到另一帧图像中的直线 l_1 上。



3D computer vision

在 3D 计算机视觉中有 2 个重要的矩阵：本质矩阵 (Essential matrix) 和基本矩阵 (Fundamental Matrix)。两者都是对场景的对极几何进行编码的 3×3 矩阵。

本质矩阵 (Essential matrix)

几何表示是便于理解的，可是计算机并不懂，我们要将其转为代数形式。我们知道由相机1到相机2是刚体运动，那么观测点 P 在相机1坐标系的坐标就可以通过刚体转换变成相机2坐标系下， $P' = RP + T$ ，其中 R 和 T 分别表示旋转和平移，如果我们将其左乘一个 T ，即 $T \times P' = T \times RP + T \times T = T \times RP$ 。其中 $T \times P'$ 表示对极平面的法线，若再左乘一个 P' 得到 $P'(T \times P') = P' \times (T \times RP)$ 。由于 P' 与法线 $T \times P'$ 是垂直的，所以有： $0 = P' \times (T \times RP)$ ，我们知道两向量的叉乘可以转换为一向量的反对称矩阵与另一向量的点乘，于是：

$P' \times (T \times RP) = P'[T \times]RP = 0$, $[T \times]$ 表示 T 的反对称矩阵，我们让 $E = [T \times]R$ ，那么 $P'E P = 0$ ，这个 E 就是本质矩阵。

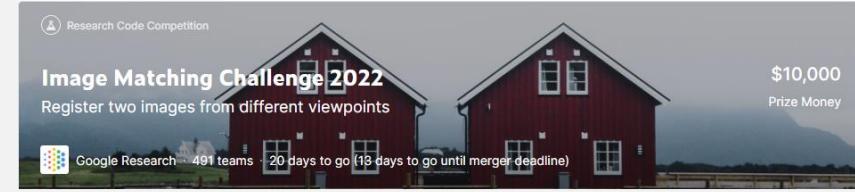
讲义2 | Find Good Correspondences



基本矩阵 (Fundamental Matrix)

基本矩阵 F 封装了同一场景的两个视图之间的投影几何。它不受场景中内容的影响，只取决于两个相机的内在和外在参数。这两个参数我们在讲义1已经介绍过了。

讲义2 | Find Good Correspondences



3D computer vision / 3D重建中最常见的问题

Triangulation: 我们知道相机参数 P (姿势) , 并且我们在图像中有点 x 。 我们可以使用以下等式 $x = PX$ 计算真实世界坐标 3D 点 X 。 (请注意, 在这种情况下 我们需要计算 X)。

- **Perspective-n-Point (PnP) 问题:** 假设我们知道 3D 维度的点 X , 并且也有这些点与图像中的 2D 点 x 之间的匹配, 如何计算相机 参数 P ? 相同的等式: $x = PX$ 但现在缺少 P 。
- **怎么计算 P ?**

我们可以将矩阵 P 视为 2D 和 3D 点之间的映射。 我们可以从 K 、 R 和 T 做到这一点! 这是我们需要的训练的一部分: $P = K [R \mid t]$, 其中 K 是 (3×3) 的矩阵, $[R \mid t]$ 是对应于 R 和最后一列的平移向量 $(3,)^T$ 。 所以 P 是一个 3×4 的矩阵。 其中 R 和 T 是相机的外部参数: 相机在世界参考系统中的位置和方向 (也称为相机姿势)。

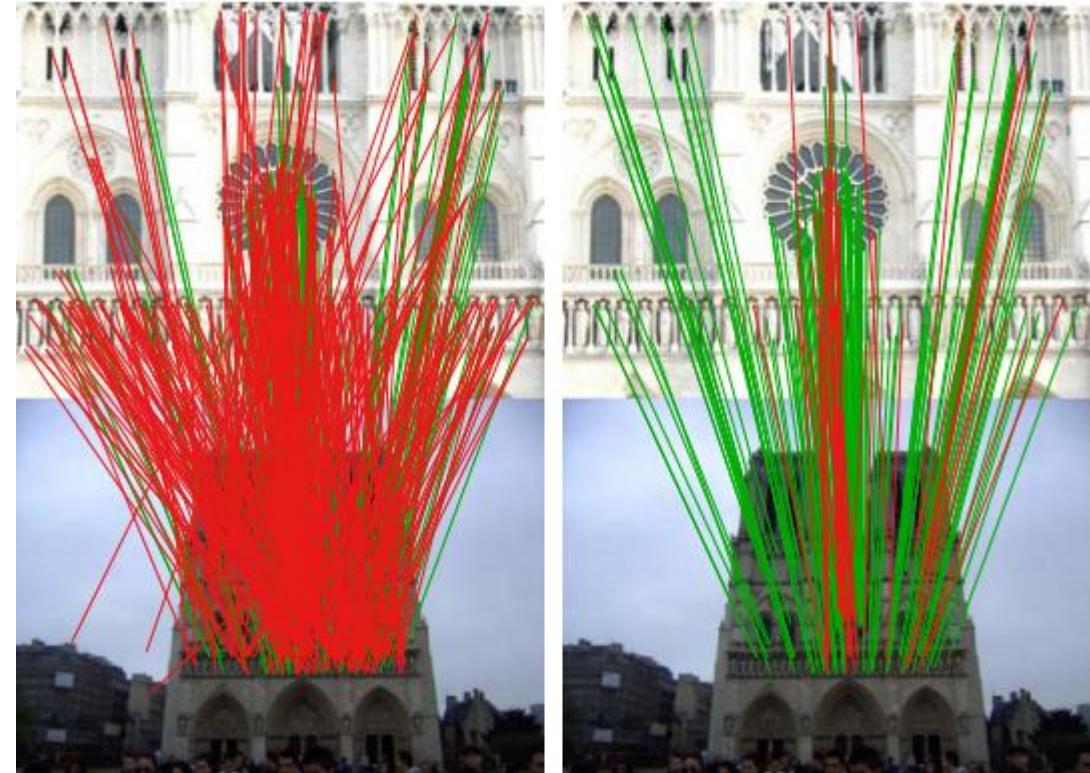
讲义2 | Find Good Correspondences



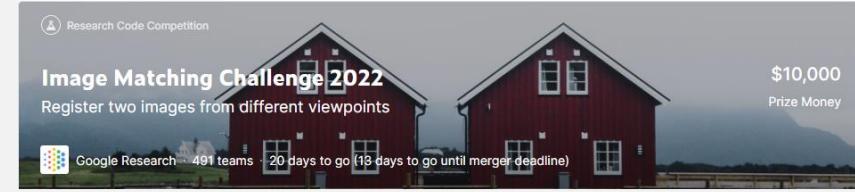
3D computer vision

解决这些问题的关键是什么？

x , 也称为关键点, 用于查找图像之间对应关系的点。可以使用 SIFT 等传统算法来获取此关键点。这里将介绍一种深度学习架构——**CNe** 能帮助我们找到更好的 Correspondences.



讲义2 | Find Good Correspondences



3D computer vision

CNe

CNe是基于像素坐标而不是直接在图像上运行的多层感知器，这也是这个架构和其他方法的最大不同之处。

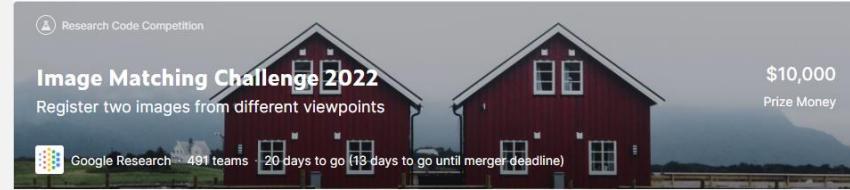
基本过程：

- 使用一组假定的稀疏匹配和相机内在函数，通过端到端的方式训练网络标记inliers和outliers。
- 通过已经标记好的nliers和outliers来还原基本矩阵 (**Fundamental Matrix**) 相对姿势。
- 引入Context Normalization将全局信息嵌入其中的同时单独处理每个数据点，并使网络对对应的顺序保持不变。

优点

- 轻量，相较于其他方法计算量较小
- 泛化性好，无论是室内还是户外都有良好的表现

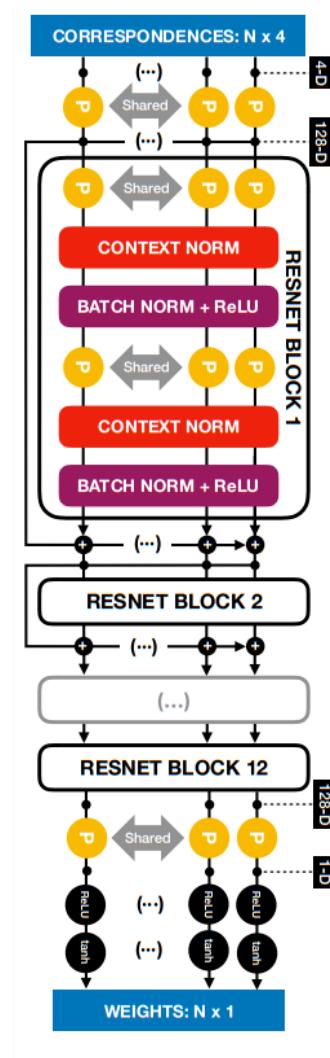
讲义2 | Find Good Correspondences



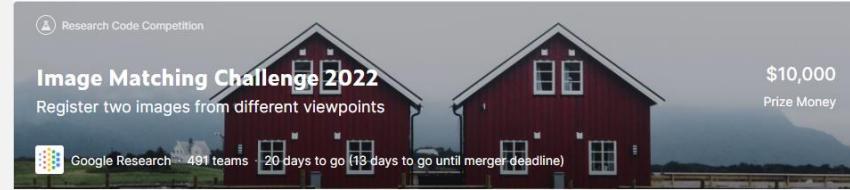
3D computer vision

CNe 通过两个图像中的特征点，以及潜在的对应关系训练了一个深度网络同时解决一个分类问题——保留或删除每Correspondences—以及一个回归问题——通过利用极线约束来检索相机运动 (camera motion)。为此 CNe的作者还引入了一种通过对 8 点算法进行简单加权重构来计算姿势的可微分方法，权重预测对应关系正确的可能性。

CNe的输入是两个2D 点之间的 N 个对应关系 ($4 \times N$ 值)，并对每个对应关系产生一个权重，将其编码为可能的inlier。每个对应由**weight-sharing Perceptrons** (P) 独立处理，使网络对输入。全局信息通过**Context Normalization**嵌入。



讲义2 | Find Good Correspondences

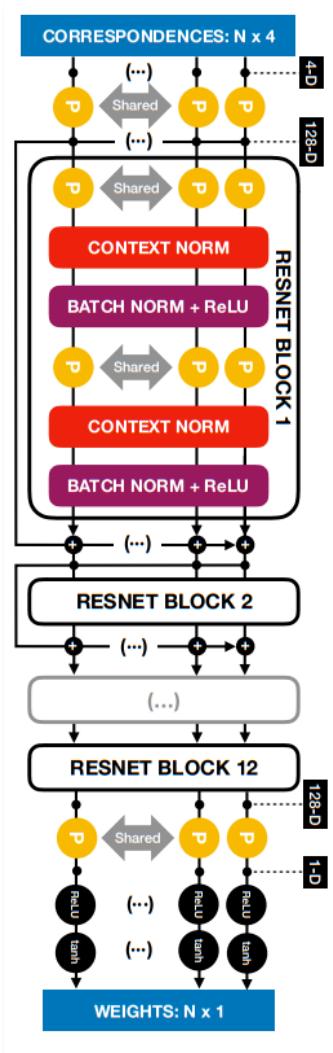


3D computer vision

Context Normalization

CN机制上类似于其他归一化方法，但作用于不同的维度。CN是对每个感知器出处的不同对correspondences进行归一化，对于每个图像对分别输出。这允许特征图的分布对场景几何和相机运动进行编码，将上下文信息嵌入到与上下文无关的 MLP 中。

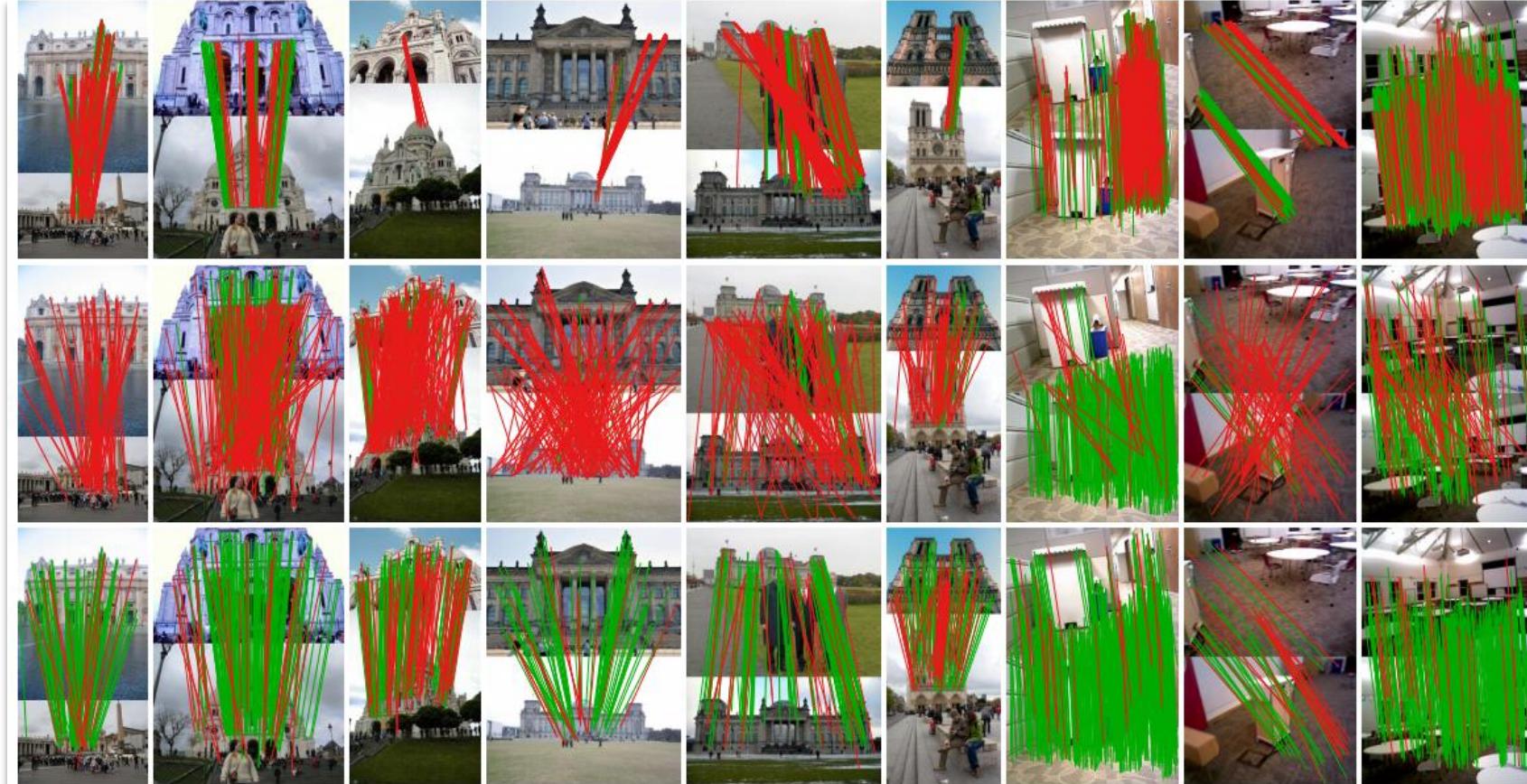
CNe的网络是一个 12 层 ResNet，其中每一层由两个残差块组成：一个 128 个神经元共享的感知机每个对应的权重，CN层、BN层和 ReLU。在最后一层感知器应用一个 ReLU 和一个 tanh 将输出限制在 $[0, 1)$ 范围内。我们使用这个截断的tanh 而不是 sigmoid，这样网络可以轻松输出 $w_i = 0$ 可以完全去除异常值。



讲义2 | Find Good Correspondences



3D computer vision | CNe的效果



讲义2 | Find Good Correspondences



Kornia: 基于PyTorch的可微计算机视觉库



The screenshot shows the Kornia library's landing page. The main title "Kornia" is displayed in large, bold, black font. Below it is the subtitle "Open Source and Computer Vision". To the right of the subtitle is a cartoon penguin wearing a blue headband with a camera lens and holding a small 3D cube. Below the penguin is the text "Powered by PyTorch". A horizontal line separates this section from the rest of the page. Below the line, the text "The open-source and Computer Vision 2.0 library" is centered. Further down, there are language links "English | 简体中文" and a navigation bar with links like "网站 · 文档 · 快速尝试 · 教程 · 例子 · 博客 · Slack社区". Below the navigation bar are several status indicators: "python 3.6 | 3.7 | 3.8", "pypi package 0.6.3", "downloads 2M", "License Apache 2.0", "Slack", "Follow @kornia_foss 3.4k", "tests-cpu failing", "tests-cuda failing", "codecov 92%", "docs passing", and "pre-commit.ci passed". At the bottom, there is a "FEATURED ON Product Hunt" badge with a "32" rating.

讲义2 | Find Good Correspondences



Kornia: 概览

概览

受现有开源库的启发，Kornia可以由包含各种可以嵌入神经网络的操作符组成，并可以训练模型来执行图像变换、对极几何、深度估计和低级图像处理，例如过滤和边缘检测。此外，整个库都可以直接对张量进行操作。

详细来说，Kornia 是一个包含以下组件的库：

Component	Description
kornia	具有强大 GPU 支持的可微计算机视觉库
kornia.augmentation	在 GPU 中执行数据增强的模块
kornia.color	执行色彩空间转换的模块
kornia.contrib	未进入稳定版本的实验性模块
kornia.enhance	执行归一化和像素强度变换的模块
kornia.feature	执行特征检测的模块
kornia.filters	执行图像滤波和边缘检测的模块
kornia.geometry	执行几何计算的模块，用于使用不同的相机模型执行图像变换、3D线性代数和转换
kornia.losses	损失函数模块
kornia.morphology	执行形态学操作的模块
kornia.utils	图像/张量常用工具以及metrics

GitHub: [kornia - Open Source Differentiable Computer Vision Library for PyTorch'](#)

讲义2 | Find Good Correspondences



Kornia: 基础使用

```
%capture
!pip install kornia

import cv2
from matplotlib import pyplot as plt
import numpy as np

import torch
import torchvision
import kornia as K

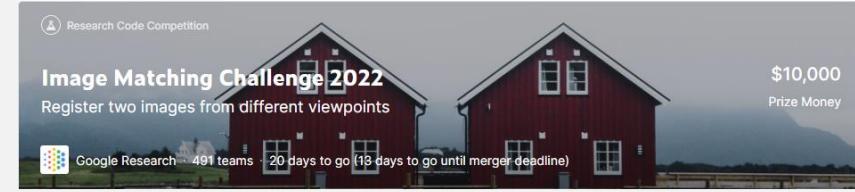
img_bgr: np.array = cv2.imread('arturito.jpg') # HxWxC / np.uint8
img_rgb: np.array = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb); plt.axis('off')

x_bgr: torch.tensor = K.image_to_tensor(img_bgr) # CxHxW / torch.uint8
x_bgr = x_bgr.unsqueeze(0) # 1xCxHxW
print(f"convert from '{img_bgr.shape}' to '{x_bgr.shape}'")

convert from '(144, 256, 3)' to 'torch.Size([1, 3, 144, 256])'
```

讲义2 | Find Good Correspondences



Kornia: 帮助理解比赛的一些代码

7-point algorithm: 使用一对的2D点的关键点作为输入

```
F, inlier_mask = cv2.findFundamentalMat(cur_kp1[match_idxs[:, 0]],  
cur_kp2[match_idxs[:, 1]], cv2.USAC_MAGSAC, ransacReprojThreshold=0.5,  
confidence=0.9, maxIters=20000)
```

从每个图像的 2k 128-D 描述符中找到匹配项

```
dists, idxs = matcher(descriptors_1[0], descriptors_2[0])
```

KeyNet 检测器 + AffNet + HardNet 描述符

```
lafs1, resp1, descriptors_1 = keynet_affnet_hardnet8(timg1)
```



讲义3

- 01 SuperGlue vs LoFTR
- 02 数据增强

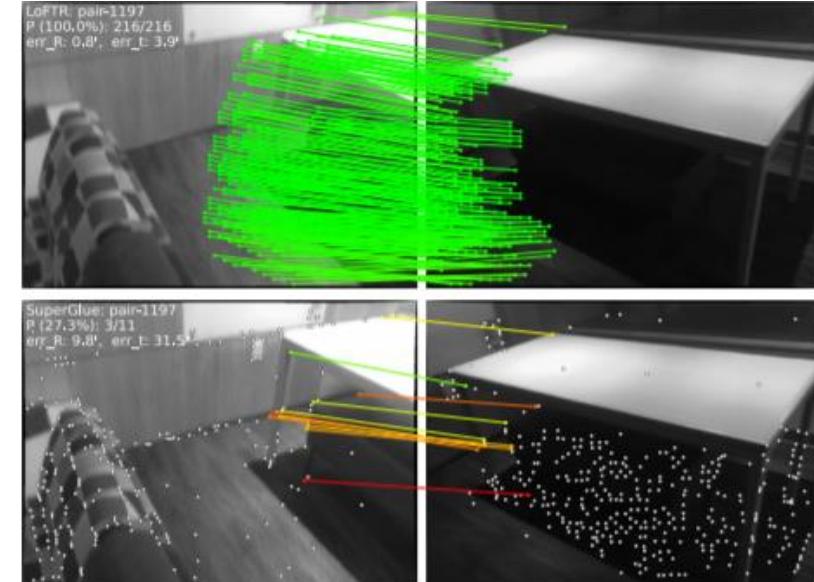
讲义3 | SuperGlue vs LoFTR



SuperGlue VS LoFTR

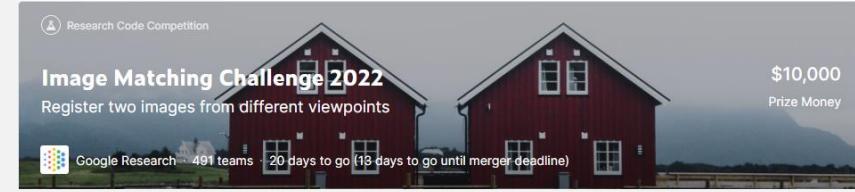
SuperGlue与**LoFTR**都是对图片间进行特征点匹配的方法，其目的是，找到图像A、图像B中同时存在的相同物体实例，并输出其位置信息、匹配关系。

在提取出特征点后通过图神经网络生成匹配代价矩阵，并求解最优匹配矩阵，以获得全局最优的匹配结果。



```
lafs1, resp1, descriptors_1 = keynet_affnet_hardnet8(timg1)
```

讲义3 | SuperGlue vs LoFTR



SuperGlue VS LoFTR

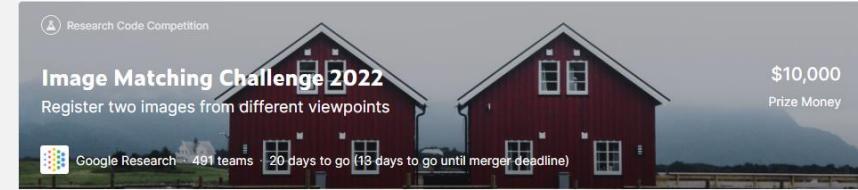
SuperGlue模型中主要采用的是基于空间方法的图注意力网络，运用了类似Vision Transformer的方法，只不过样本空间并不是全图像的Patch Embedding，而是提取所得的特征点编码信息（位置+由传统方法或深度卷积方法获得的该像素及其邻域的特征+特征置信度）。图中的每个数据样本（节点）是特征点的编码信息，每条边是节点两两之间的Attention值，每个节点都有边与其他所有节点连通，构成了一个完全图。我们通过下式计算完全图的边信息。

$$\text{Attention} = \sum_{j:(i,j) \in \varepsilon} [\text{Softmax}_j(\mathbf{q}_i \mathbf{k}_j^\top)] \mathbf{v}_j$$

这是一种类似于数据库检索的特征聚合方式，在图神经网络中也被称为“消息传递”。 \mathbf{q} 表示Query， \mathbf{k} 表示Key， \mathbf{v} 表示Value，对于图像对{A, B}， \mathbf{q} 由A产生， \mathbf{k} 、 \mathbf{v} 由B产生。对于输入A，我们通过 \mathbf{q} 进行查询，匹配到B的键 \mathbf{k} ，通过Softmax函数计算这种查询与键匹配的相似度，然后与B的真值 \mathbf{v} 相乘。通常这种Attention机制会循环多层，以使匹配达到紧密收敛。需要注意的是，对于每轮循环，我们首先对{A, A}、{B, B}计算Self-Attention，再对{A, B}计算Cross-Attention。

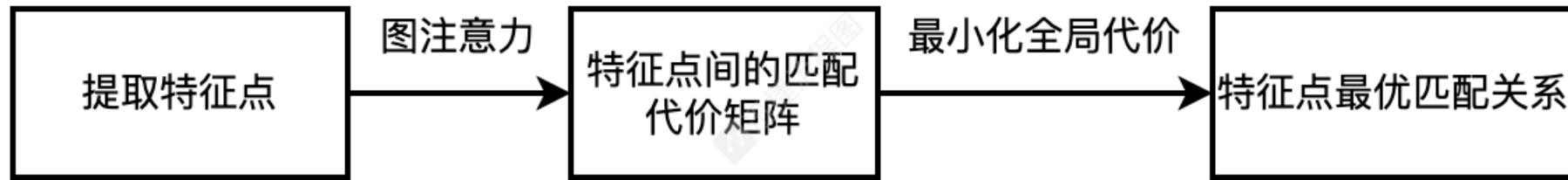
成功的编码表示了特征点，并计算出特征点间的互相关性。**就可以将这两者相加融合，这相当于将完全图中所有的边近似去除，使所有节点之间互相独立**，此时我们简单计算节点间的内积（欧氏距离）即可进行后续的匹配流程。

讲义3 | SuperGlue vs LoFTR

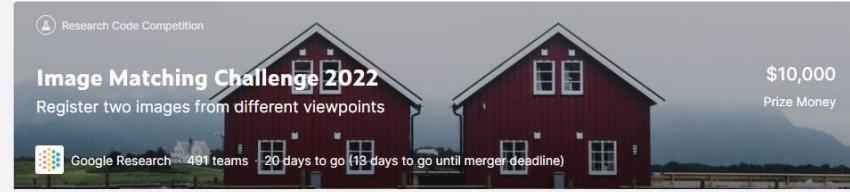


SuperGlue

SuperGlue处理一对图像对的主干逻辑



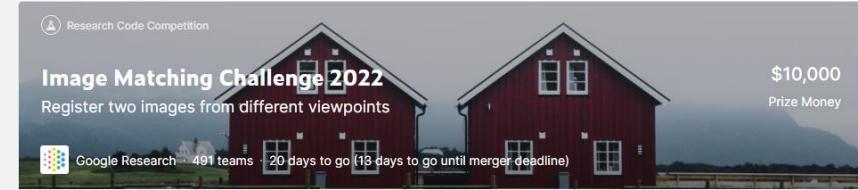
讲义3 | SuperGlue vs LoFTR



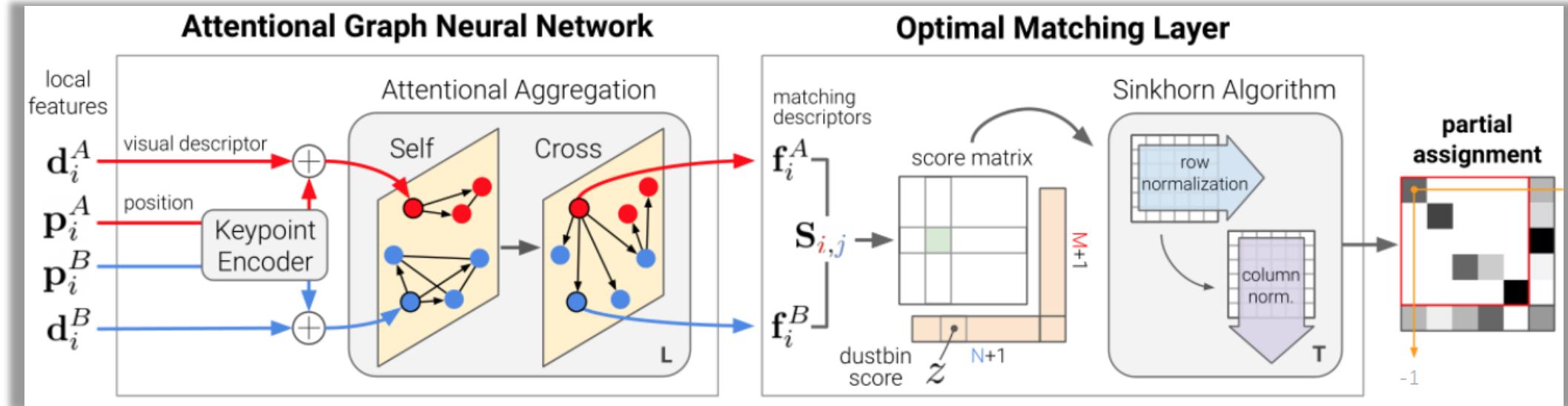
SuperGlue | 模型训练阶段

- 输入1张真实图片，提取特征点，通过随机单应变换生成1张特征点对应的新图片、以及深度图，形成一对图像对。
- 深度图生成的方法是：使用COLMAP/MVS方法，对一些具有多视角多图的场景进行稀疏-稠密重建，然后对生成图进行深度估计。
- 原文给出的预训练模型采用Oxford、Paris数据集做单应估计训练分别采用百万级的两个数据集Scannet数据集（室内）、YFCC100M数据集（室外）做姿态估计训练。
- SuperGlue原文链接：[SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems](#)

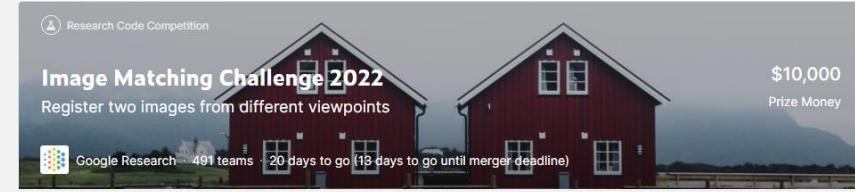
讲义3 | SuperGlue vs LoFTR



SuperGlue | 模型流程



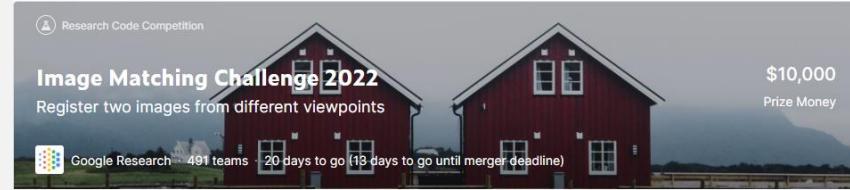
讲义3 | SuperGlue vs LoFTR



SuperGlue | 模型流程

1. Keypoint Encoder模块对每个特征点进行编码，将特征描述子d与位置信息p合成为一个向量。
2. 图像A含有N个特征点，表示为一个N长度序列。图像B含有M个特征点，表示为一个M长度序列。接下来构建含N+M个节点的完全图，每个节点表示一个特征点，每条边表示两个特征点之间的关系，这种关系是通过Attention机制实现的。
3. 经过多轮Attention之后，每个特征点有了一个聚合表示。通过对聚合表示两两内积可得N*M尺寸的代价矩阵，这表示了将特征点两两匹配所需付出的代价。如果两个点非常不相似，如果强行把他们匹配，则会付出很大的代价。
4. 基于代价矩阵，需要线性规划解算一个最优分配矩阵，其目标是，以最小的全局代价两两匹配图像A与图像B中的特征点，但其计算复杂度随数据量增长而变大。所以采用SinkHorn算法，添加熵正则化约束以近似求解。这里需要注意，对于无法越过阈值成功匹配的点，使用额外的垃圾桶空间来存储这些点，为最优化过程添加约束，从而滤除错误点。SinkHorn算法在保证按需分配的同时，由于熵正则化的作用会使分配矩阵偏向均匀分布，因此结果会有一定的平滑。
5. 模型中使用了许多简易的多层感知机，这减少了复杂计算，调节了通道尺寸，并添加了非线性以提高抽象表征能力。
6. 损失函数为匹配代价矩阵的负对数似然损失。这相当于优化匹配的准召。

讲义3 | SuperGlue vs LoFTR



SuperGlue | 模型推理阶段:

输入：包含特征点位置信息、特征描述子，可通过手工特征或深度学习提取。

手工：SIFT算法

深度：SuperPoint模型

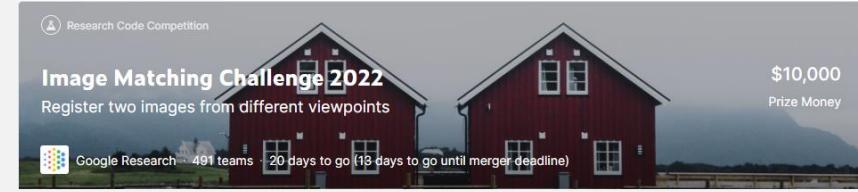
输出：2张图得特征点间得最优匹配关系。

设定阈值输出结果。

阈值包括：可作为特征点的置信度阈值、是否输出该匹配的匹配得分阈值

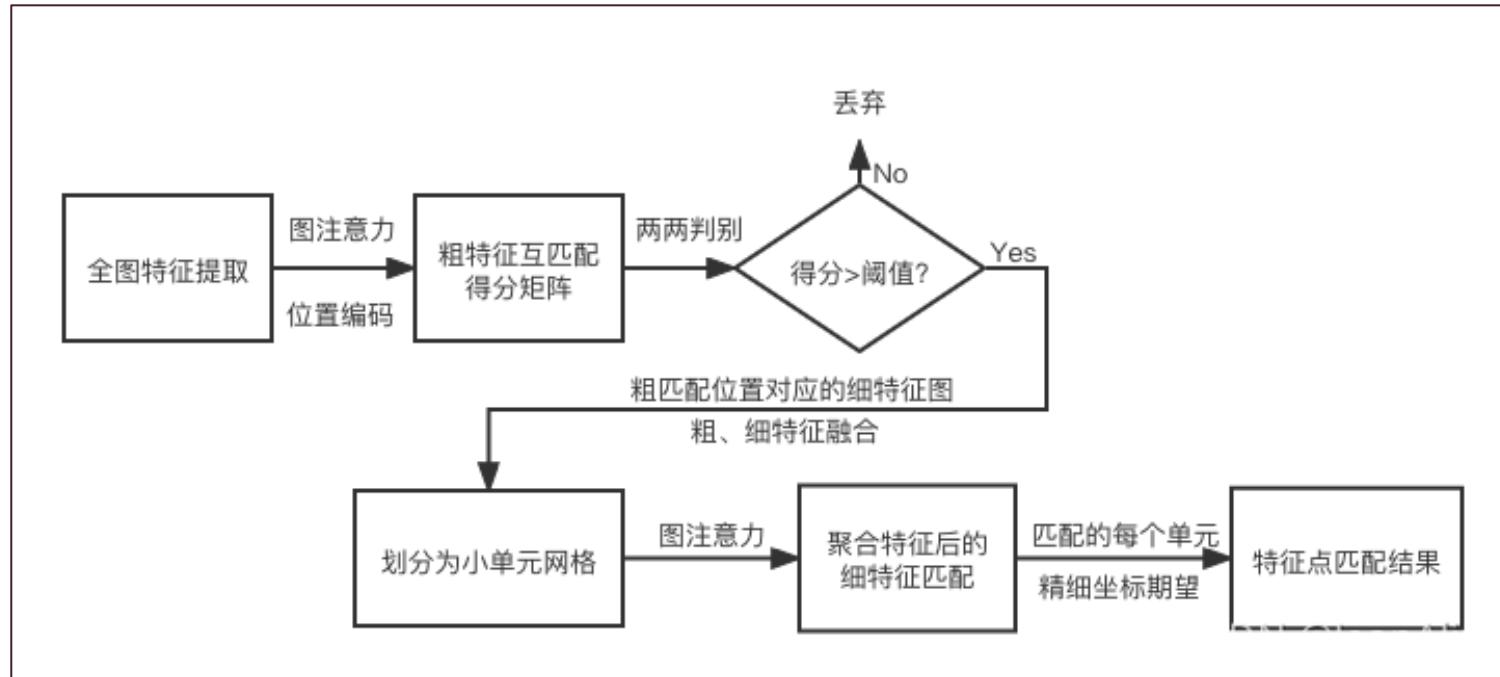


讲义3 | SuperGlue vs LoFTR

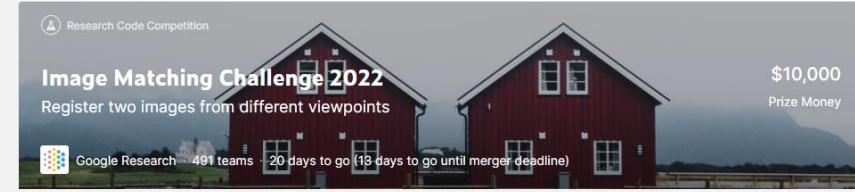


LoFTR

LoFTR处理一对图像对的主干逻辑



讲义3 | SuperGlue vs LoFTR



LoFTR

LoFTR基于SuperGlue的最主要的改进是：

- SuperGlue基于预先提取的部分特征点。而LoFTR则像传统的图像识别方法那样，预先对全图提取特征，并在将特征送入Transformer前，为每个点添加位置编码。
- SuperGlue的特征描述子尺度是不变的，LoFTR先对较大的图像块之间做粗匹配，对于可信配对，再做进一步的精细高分辨率匹配。
- LoFTR原论文链接：[LoFTR: Detector-Free Local Feature Matching with Transformers](#)

讲义3 | SuperGlue vs LoFTR

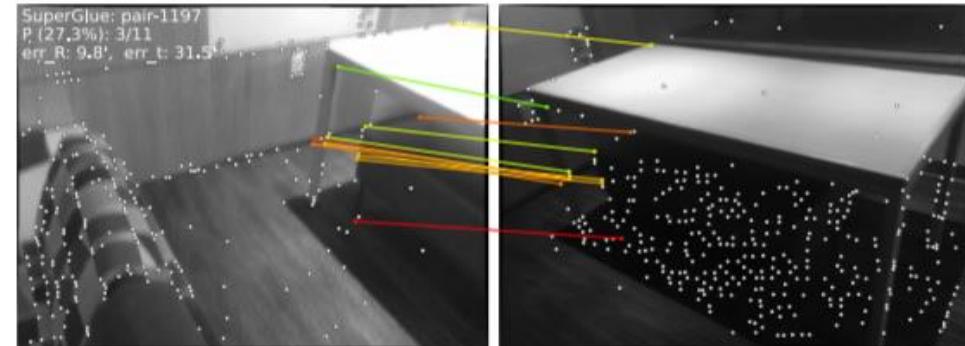
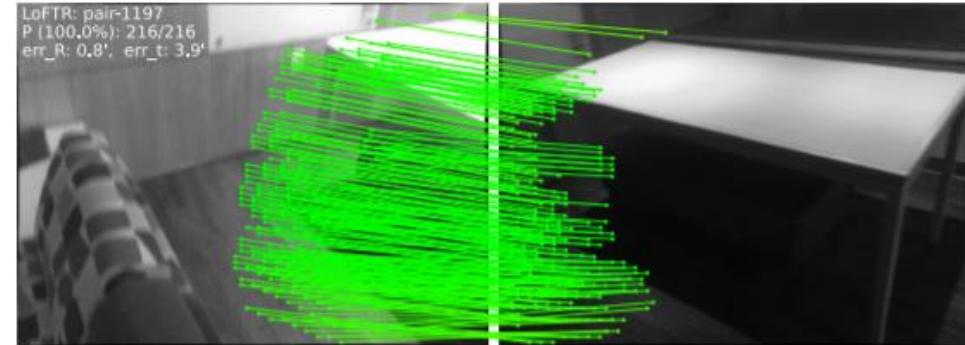


LoFTR

相较于SuperGlue，LoFTR主要解决了两个关键问题：

1. LoFTR可以显著提升低/无纹理点的配准

如右图所示，LoFTR可以在无纹理的墙壁和地板上找到重复图案的对应，而这对于SuperGlue是不可实现的。



讲义3 | SuperGlue vs LoFTR

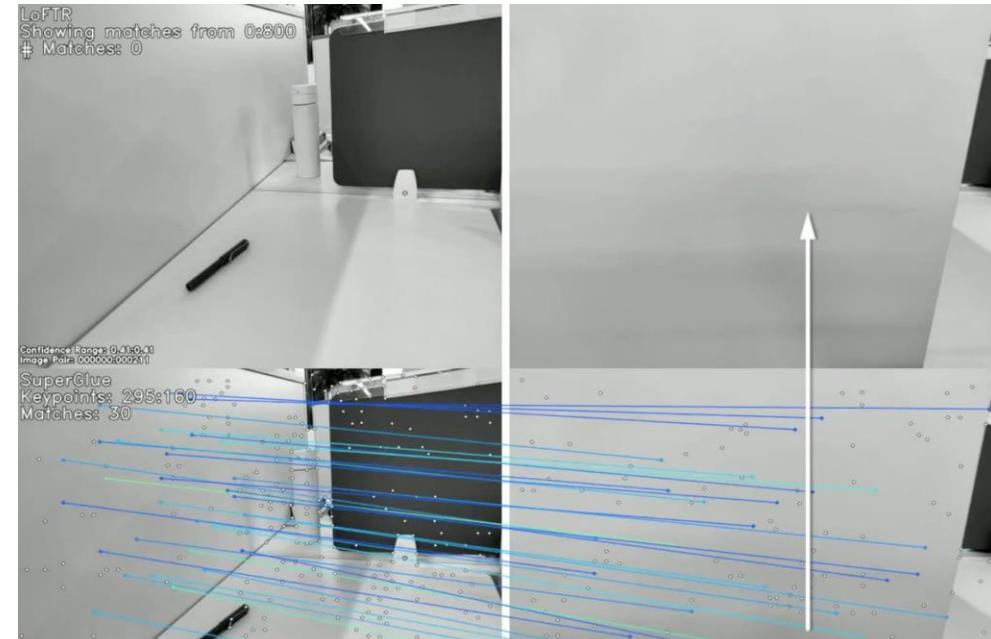


LoFTR

相较于SuperGlue，LoFTR主要解决了两个关键问题：

2. 在处理具有近似纹理特征点时，LoFTR更为谨慎、准确，对于图像块所属的实例关系判断更准确。

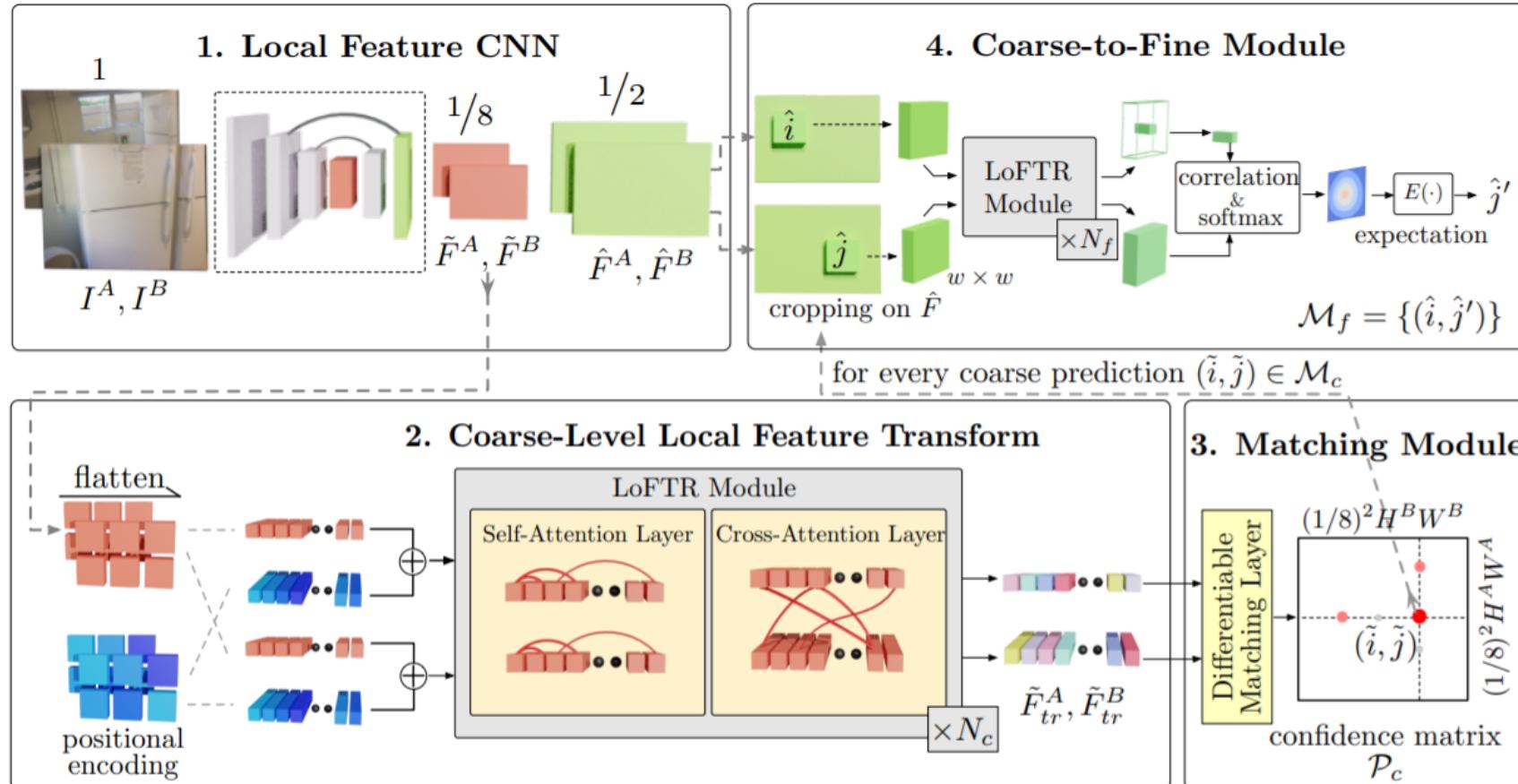
如右图所示，这是从右侧、左侧匹配桌子场景的结果。显然 SuperGlue对于墙壁上的点产生了误匹配，而LoFTR则谨慎的多。



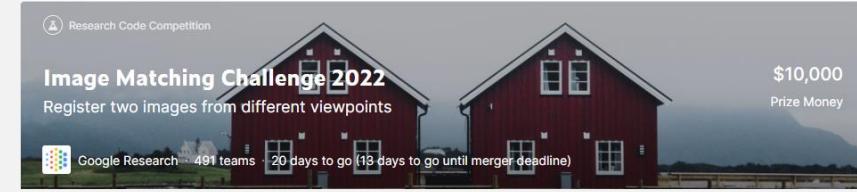
讲义3 | SuperGlue vs LoFTR



LoFTR | 模型流程



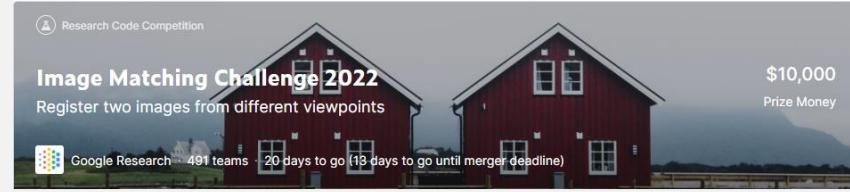
讲义3 | SuperGlue vs LoFTR



LoFTR | 模型流程

1. 局部特征提取网络从图像对{A, B}中提取粗特征图、精特征图。
2. 将A, B的粗略特征图分别展平为一维序列，并分别融合位置编码；然后，对两序列计算图注意力（输入Transformer），通过SinkHorn算法或Dual Softmax法，得到粗特征块互匹配得分矩阵。根据置信度阈值、相互邻近标准，得到粗匹配预测。
3. 对于粗匹配预测对应的A, B中的位置，我们分别将其对应的细特征图按 5×5 单元划分为网格图，并序列化，对于每个单元，都将其细特征与网格中心点粗特征融合，得到了每个单元的细特征表示。
4. 对于A, B的细特征序列计算图注意力，对于两两配对，我们以A的单元中心点为一个最终特征点坐标，输出B的 5×5 单元中置信度的坐标。最终，我们选择置信度最高的top_K个匹配输出。
5. 损失函数包含两部分：粗匹配互相关得分矩阵的负对数似然损失 + 细匹配坐标L2损失。这相当于优化粗匹配的准召，以及细匹配位置的准确率。

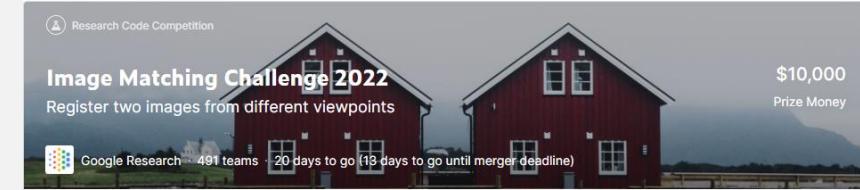
讲义3 | SuperGlue vs LoFTR



LoFTR | 模型训练阶段：

1. 属于监督训练，无需手工标注，自标注生成方法采用类似SuperGlue的随机单应变换法，生成图像对及深度图。
2. 不同之处在于，在选定标注时，为了避免出现一对多的情况，遵循互最近邻原则；对于粗标签，先由左向右变换，取右侧最近邻点作为右侧gt标签候选，再将该候选点由右向左变换，取左侧最近邻点，如果左侧最近邻点与初始锚点相同，则该左右点对作为一个gt点对。对于细标签，我们将左侧gt粗标签的中心点变换到右侧，取细尺度的最近邻点作为右侧候选点，对于一张变换生成的标签图像，粗标签的中心点必须落在对应生成的最近邻细标签的 5×5 单元窗口内，否则丢弃该粗标签。
3. 原文给出的预训练模型使用ResNet-18作为backbone特征提取器，分别采用百万级的两个数据集Scannet数据集（室内）、MegaDepth数据集（室外）做姿态估计训练。

讲义3 | SuperGlue vs LoFTR

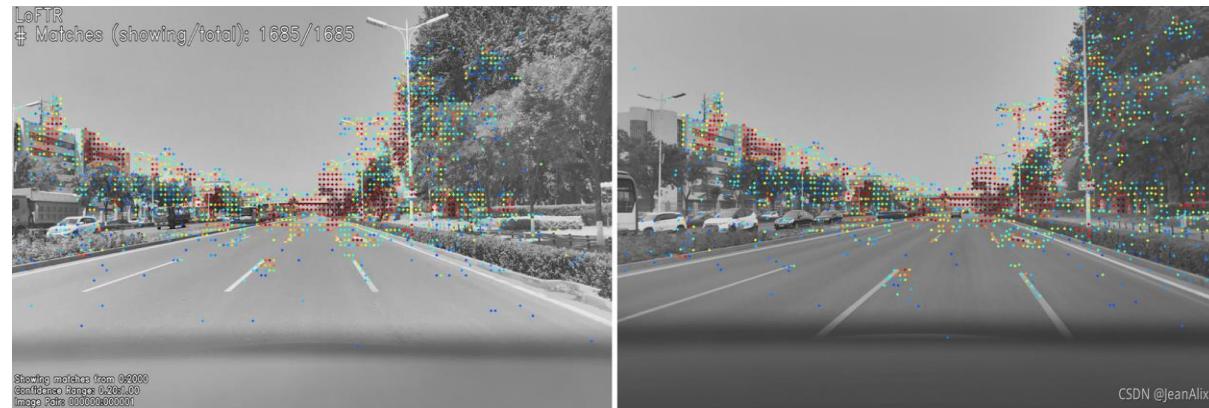


LoFTR | 模型推理阶段

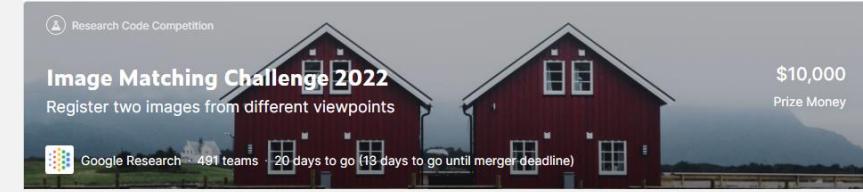
输入：1个图像对

由CNN逐像素自动提取的多尺度特征+位置编码

输出：2张图的特征点间的最优匹配关系。



讲义3 | SuperGlue vs LoFTR



SuperGlue vs LoFTR

SuperGlue源代码：[SuperGlue PyTorch Implementation](#)

LoFTR源代码：[LoFTR: Detector-Free Local Feature Matching with Transformers](#)

讲义3 | 数据增强



数据增强

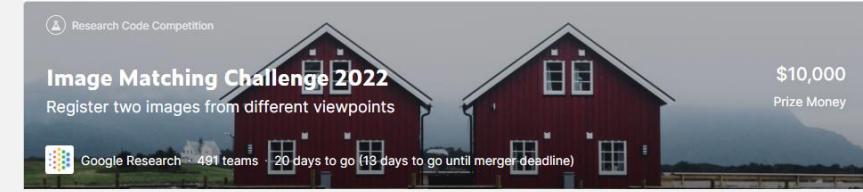
我们常常会遇到数据不足的情况。

比如，你遇到的一个任务，目前只有小几百的数据，然而现在流行的最先进的神经网络都是成千上万的图片数据。

我们可以通过 **数据增强**，来变相地增加图片数量。

本次竞赛中，训练集只有5600多张图片，所以通过数据增强来增加图片数量非常重要。并且起到了防止过拟合的效果。

讲义3 | 数据增强

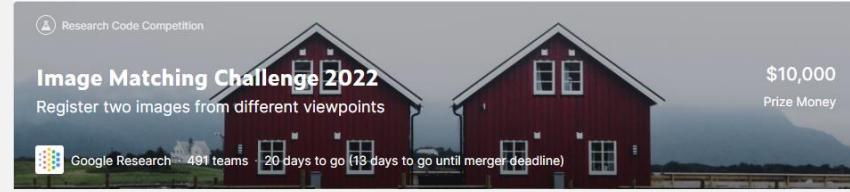


常见的数据增强

几何变换：翻转，旋转，裁剪，变形，缩放等各类操作。

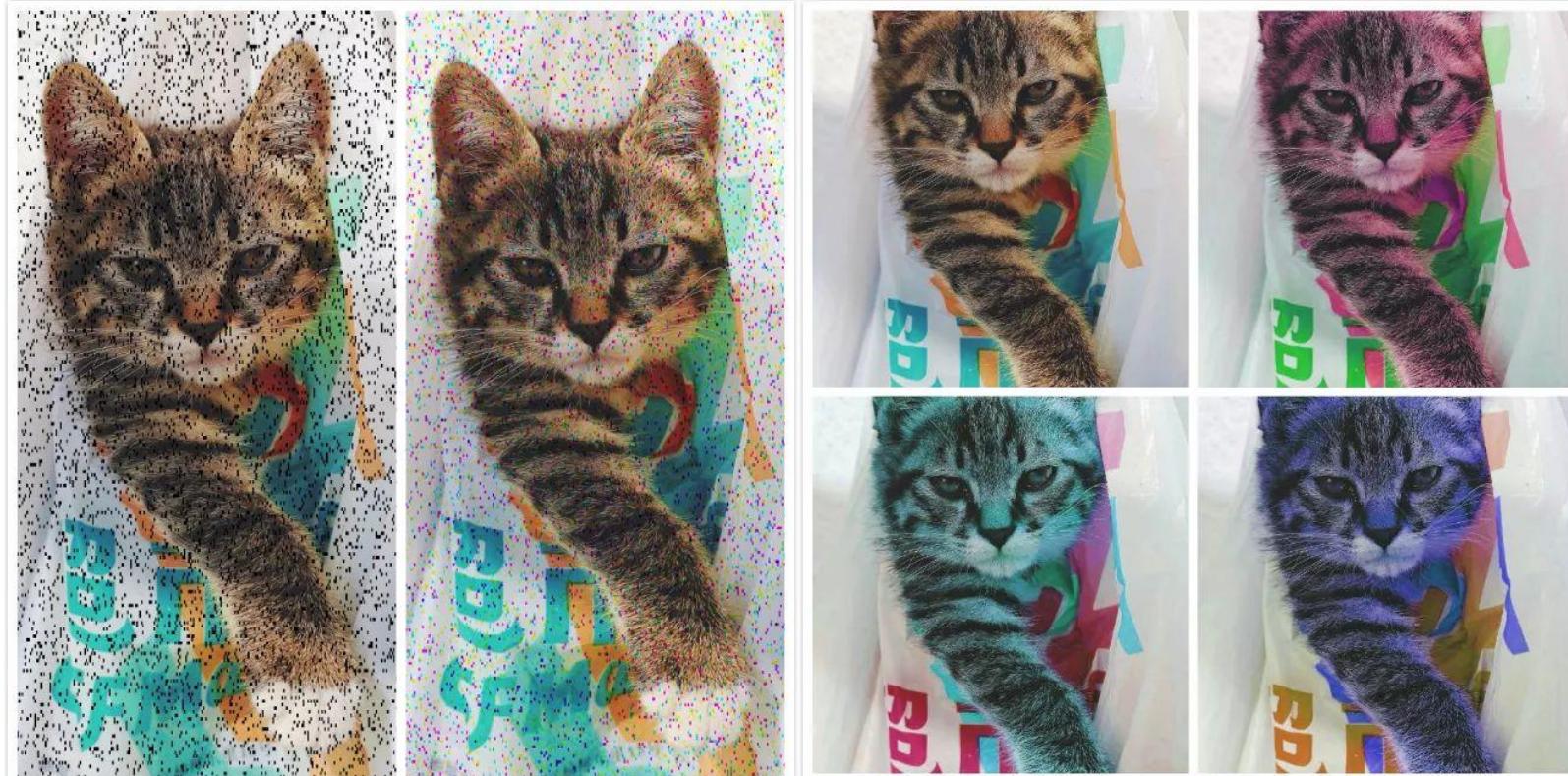


讲义3 | 数据增强

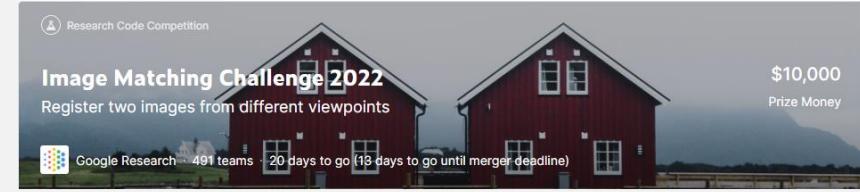


常见的数据增强

颜色变换类：常见的包括噪声、模糊、颜色变换、擦除、填充等等。

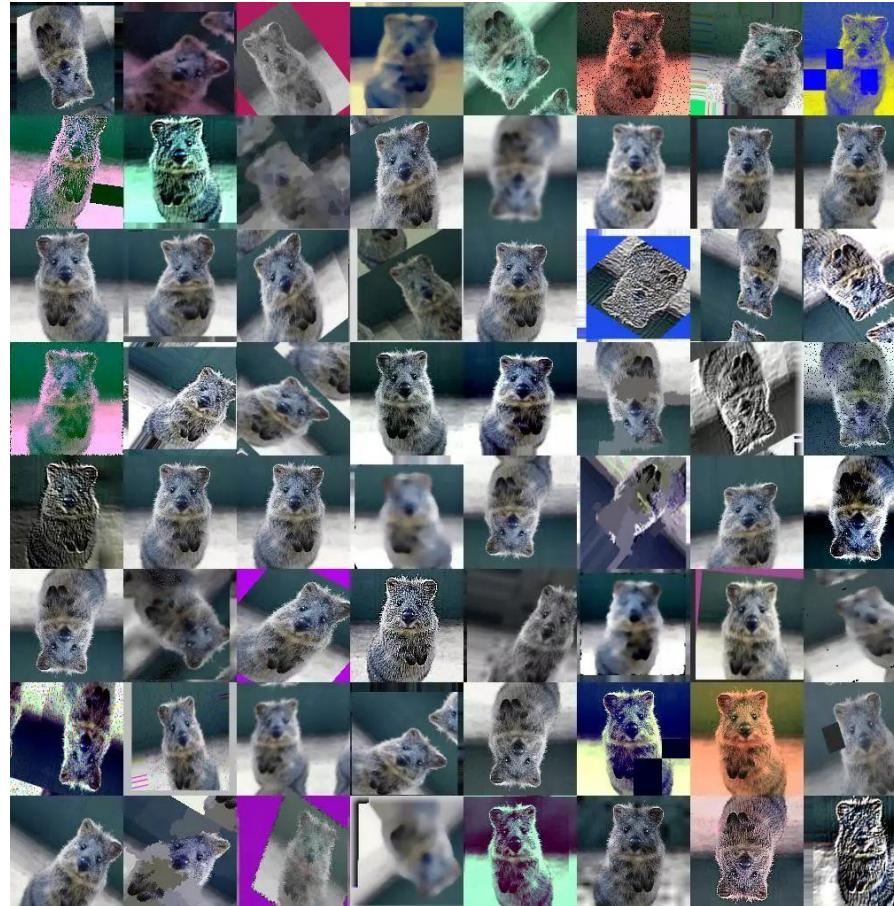


讲义3 | 数据增强



常见的数据增强

混合前两种方法



讲义3 | 数据增强



本次竞赛数据增强

我目前使用了以下数据增强，都是相对比较简单和常规的，**并且本次比赛对颜色类的增强效果不好。**

SmallestMaxSize 设置图片最短边的最大值

RandomCrop 图片随机裁剪

VerticalFlip 图片垂直翻转

HorizontalFlip 图片水平翻转

Normalize 归一化 让RGB三色的数值分布接近，可以使得训练更稳定

讲义3 | 数据增强



本次竞赛数据增强

总之，这次竞赛较难训练，所以对各种超参都比较敏感。其中包括对数据增强的参数也非常敏感。
后续我也还会对数据增强继续调参和其他数据增强方式的尝试。

讲义4

- MAGSAC++
- Matchfomer
- 模型融合



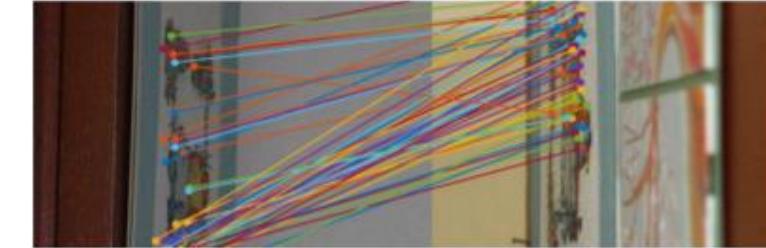
讲义4 | MAGSAC++

MAGSAC++

MAGSAC++ 引入了一个新的模型质量（评分）函数，它不需要内外点决策，以及一个新的边缘化过程，它被表述为一个带有一类新的 M 估计器（一个健壮的内核）的估计，通过迭代重新加权最小来解决 平方程序。我们还为类似 RANSAC 的稳健估计器提出了一个新的采样器 Progressive NPSAC。利用附近点通常来自真实世界数据中的相同模型这一事实，它比全局采样器更早地找到局部结构。从局部采样到全局采样的渐进过渡不会受到纯局部采样器的弱点的影响。在用于单应性和基本矩阵拟合的六个公开可用的真实世界数据集上，MAGSAC++ 产生的结果优于最先进的稳健方法。它速度更快，几何精度更高并且很少失败。



(a) Community Photo Collection dataset [30].



(b) ExtremeView dataset [11].



(c) Tanks and Temples dataset [10].



讲义4 | MAGSAC++

MAGSAC++

RANSAC算法是计算机视觉领域最广泛使用的鲁棒估计器。RANSAC重复选择输入点集的最小随机子集，并拟合模型。RANSAC算法自发布以来有许多改进的算法，如：

- MLESAC：通过最大似然过程的有益特性估计模型质量。在实践中，MLESAC结果通常优于RANSAC，且对用户定义的内点-异常值阈值不敏感。
- MINPRAN：减少对于外点内点阈值的依赖。
- MAGSAC：边缘样本共识，使用噪声参数估计模型阈值。除了不需要估计阈值，MAGSAC在其他方面也比其它估计器更加稳健。用加权最小二乘替代原先的最小二乘，权重通过边缘化程序计算。

但是MAGSAC比其他估计器慢许多，所以在MAGSAC++中提出一个新的质量和模型函数，迭代重新加权最小二乘程序求解的一类新型 M 估计器（稳健核）。提出的MAGSAC++比原始方法更准确，并快一个数量级。



讲义4 | MAGSAC++

MAGSAC++ 基于两个基本假设

噪声等级 σ 是具有密度函数 $f(\sigma)$ 的随机变量，没有先验信息，且 σ 服从均匀分布。

对于 σ ，内点的误差是由 n 自由度的X分布和 σ 的密度函数决定的，当 $r < (\sigma)$ 时：

$$g(r | \sigma) = 2C(n)\sigma^{-n} \exp(-r^2/2\sigma^2)r^{n-1},$$

当 $r > \tau(\sigma)$ 时， $g(r | \sigma) = 0$ ，对比度 $C(n) = (2^{n/2}\Gamma(n/2))^{-1}$ ，对于 $a > 0$ ，有

$$\Gamma(a) = \int_0^{+\infty} t^{a-1} \exp(-t) dt$$

$\Gamma(\cdot)$ 为 Γ 函数。

对于每一个可能的值 σ ，点 p 属于内点集 P 的概率为：

$$\mathbf{P}(p | \theta_\sigma, \sigma) = 2C(n)\sigma^{-n} D^{n-1}(\theta_\sigma, p) \exp\left(\frac{-D^2(\theta_\sigma, p)}{2\sigma^2}\right)$$

其中 $D(\theta_\sigma, p)$ 为点到模型的误差



讲义4 | MAGSAC++

MAGSAC++ 基于两个基本假设

在MAGSAC++中提出一个新方法，在边缘化噪声水平 σ 时需要几个 LS 拟合。

所提出的算法是一种迭代重加权最小二乘法 (IRLS)，其中第 $(i + 1)$ 步中的模型参数计算如下：

$$\theta_{i+1} = \arg \min_{\theta} \sum_{p \in \mathcal{P}} w(D(\theta_i, p)) D^2(\theta, p), \quad (1)$$

where the weight of point p is

$$w(D(\theta_i, p)) = \int P(p | \theta_i, \sigma) f(\sigma) d\sigma \quad (2)$$

and $\theta_0 = \theta$, i.e., the initial model from the minimal sample.

为了能够选出最能解释数据的模型，质量函数Q定义如下：

$$Q(\theta, \mathcal{P}) = \frac{1}{L(\theta, \mathcal{P})}, \quad (3)$$

where

$$L(\theta, \mathcal{P}) = \sum_{p \in \mathcal{P}} \rho(D(\theta, p)),$$



讲义4 | MAGSAC++

MAGSAC++

在**MAGSAC++**中同样也改进了**NAPSAC**算法，提出了一种新的采样技术，它逐渐从局部移动到全局，假设最初局部化的最小样本更有可能是全内的。如果假设不导致终止，则该过程逐渐向 RANSAC 的随机抽样移动。

NAPSAC算法主要步骤如下：

1. 随机从所有点中选择初始点；
2. 以为中心找到其半径为的超平面的点集；
3. 若中点的数量少于最小采样值，重复步骤1；
4. 点和形成最小一致样本。

在实践中，局部抽样存在三个主要问题。首先，适合局部所有内点样本的模型通常太不精确。其次，在某些情况下，从本地样本估计模型会导致退化。例如，对于基本矩阵拟合，对应关系必须来自多个平面。这通常意味着远距离的对应是有益的。因此，纯局部采样行不通。第三，当具有全局结构时，例如图像对中背景的刚性运动，局部采样比全局采样慢得多。因此，作者建议在局部和全局采样之间进行过渡，逐渐从一种混合到另一种。



讲义4 | MAGSAC++

MAGSAC++ 改进NAPSAC算法

Algorithm 1 Outline of Progressive NAPSAC.

Input: \mathcal{P} – points; \mathcal{S} – neighborhoods; n – point number

1: $t_1, \dots, t_n := 0$ ▷ The hit numbers.

2: $k_1, \dots, k_n := m$ ▷ The neighborhood sizes.

Repeat until termination:

Selection of the first point:

3: Let \mathbf{p}_i be a random point. ▷ Selected by PROSAC.

4: $t_i := t_i + 1$ ▷ Increase the hit number.

5: **if** ($t_i \geq T'_{k_i} \wedge k_i < n$) **then**

6: $k_i := k_i + 1$ ▷ Enlarge the neighborhood.

Semi-random sample \mathcal{M}_{i,t_i} of size m :

7: **if** $\mathcal{S}_{i,k_i-1} \neq \mathcal{P}$ **then**

8: Put \mathbf{p}_i ; the k_i th nearest neighbor; and $m-2$ random points from \mathcal{S}_{i,k_i-1} into sample \mathcal{M}_{i,t_i} .

9: **else**

10: Select $m-1$ points from \mathcal{P} at random.

Increase the hit number of the points from \mathcal{M}_{i,t_i} :

11: **for** $\mathbf{p}_j \in \mathcal{M}_{i,t_i} \setminus \mathbf{p}_i$ **do** ▷ For all points in the sample,

12: **if** $\mathbf{p}_i \in \mathcal{S}_{j,k_j}$ **then** ▷ if the i th one is close,

13: $t_j := t_j + 1$ ▷ increase the hit number.

Model parameter estimation

14: Compute model parameters θ from sample \mathcal{M}_{i,t_i} .

Model verification

15: Calculate model quality.



讲义4 | MAGSAC++

MAGSAC++

MAGSAC++论文链接: [MAGSAC++, a fast, reliable and accurate robust estimator](#)

MAGSAC++代码链接: [The MAGSAC and MAGSAC++ algorithms for robust model fitting without using a single inlier-outlier threshold](#)



讲义4 | Optuna 自动调参神器

小Demo

- Optimize hyperparameters of the model

The hyperparameters of the above algorithm are `n_estimators` and `max_depth` for which we can try different values to see if the model accuracy can be improved. The `objective` function is modified to accept a trial object. This trial has several methods for sampling hyperparameters. We create a study to run the hyperparameter optimization and finally read the best hyperparameters.

```

1 import optuna
2
3 def objective(trial):
4     iris = sklearn.datasets.load_iris()
5
6     n_estimators = trial.suggest_int('n_estimators', 2, 20)
7     max_depth = int(trial.suggest_float('max_depth', 1, 32, log=True))
8
9     clf = sklearn.ensemble.RandomForestClassifier(
10         n_estimators=n_estimators, max_depth=max_depth)
11
12     return sklearn.model_selection.cross_val_score(
13         clf, iris.data, iris.target, n_jobs=-1, cv=3).mean()
14
15 study = optuna.create_study(direction='maximize')
16 study.optimize(objective, n_trials=100)
17
18 trial = study.best_trial
19
20 print('Accuracy: {}'.format(trial.value))
21 print("Best hyperparameters: {}".format(trial.params))

```



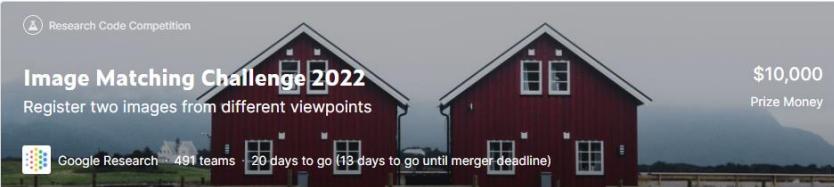
讲义4 | Optuna 自动调参神器

在optuna里最重要的三个term：

- (1) Trial：对objective函数的一次调用；
- (2) Study：一个优化超参的session，由一系列的trials组成；
- (3) Parameter：需要优化的超参；在optuna里，study对象用来管理对超参的优化，`optuna.create_study()`返回一个study对象。

optuna支持很多种搜索方式：

- (1) `trial.suggest_categorical('optimizer', ['MomentumSGD', 'Adam'])`：表示从SGD和adam里选一个使用；
- (2) `trial.suggest_int('num_layers', 1, 3)`：从1~3范围内的int里选；
- (3) `trial.suggest_uniform('dropout_rate', 0.0, 1.0)`：从0~1内的uniform分布里选；
- (4) `trial.suggest_loguniform('learning_rate', 1e-5, 1e-2)`：从 $1e-5 \sim 1e-2$ 的log uniform分布里选；
- (5) `trial.suggest_discrete_uniform('drop_path_rate', 0.0, 1.0, 0.1)`：从0~1且step为0.1的离散uniform分布里选；



讲义4 | Optuna 自动调参神器

本次比赛中，
我们可以用 **fastai** 或者 **pytorch LoFTR**
中的重要参数进行调参



讲义4 | Matchformer

Matchformer

匹配同一场景的两个或多个视图是许多基本计算机视觉任务的核心，例如，SfM、SLAM、相对姿态估计和视觉定位等。对于基于视觉的匹配，经典的基于检测器的方法，加上手工制作的局部特征，由于局部特征的高维，计算量很大。最近基于深度学习的工作侧重于使用卷积神经网络 (CNN) 学习检测器和局部描述符。然而，CNN 难以捕捉全局表达，例如视觉元素之间的远距离关系。

近期，Transformer 架构 已被引入许多视觉任务，它可以捕捉复杂的空间变换和长距离特征依赖关系，形成有利于纹理稀疏场景中特征匹配的全局表示。

实验证明更高的分辨率对于局部特征匹配更好。但是自注意力的计算对于高分辨率来说是非常昂贵的，所以LoFTR和SuperGlue仅在解码器顶部应用注意力模块。使解码器负担过重，却忽略了编码器的匹配能力，使整个模型的计算效率低下。



讲义4 | Matchformer

Matchformer

为了提高计算效率和匹配低纹理场景的鲁棒性，

MatchFormer的作者提出了**interleaving self- and cross-attention**来构建一个匹配感知编码器，这样可以同时学习图像本身的局部特征及其配对图像的相似性，即提取和匹配，从而减轻了解码器的过重，提高了整个模型的效率。在编码器早期阶段安排的交叉注意力增强了特征匹配，特别是在低纹理室内场景或室外训练样本较少的情况下。

为了提取连续的补丁信息并嵌入位置信息，作者在匹配感知编码器中设计了一种新的位置补丁嵌入（PosPE）方法，可以增强对低级特征的检测。

下图是Matchfomer和LoFTR的对比

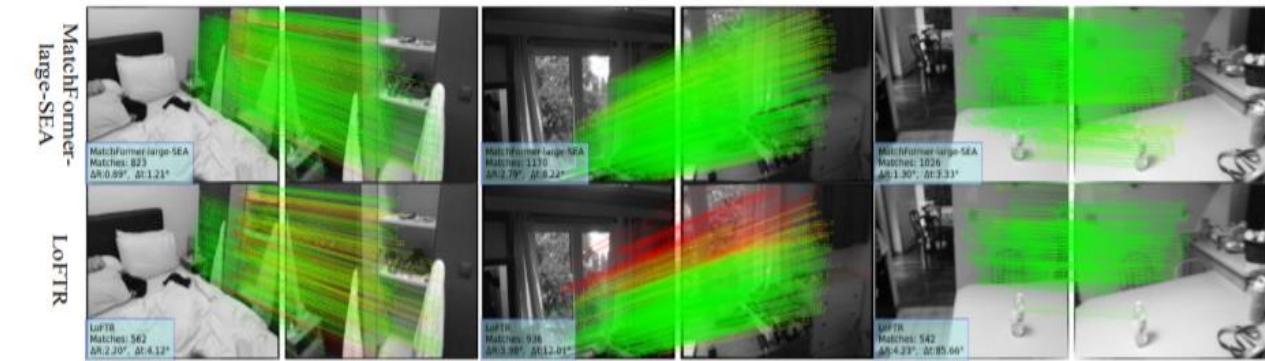


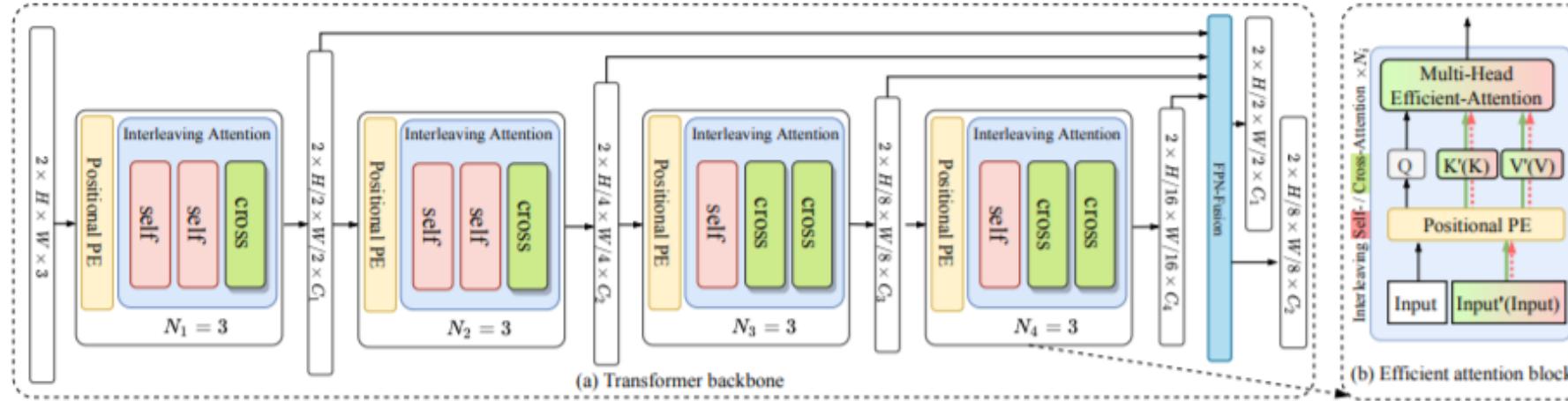
Figure 5. Qualitative visualization of MatchFormer and LoFTR [33]. MatchFormer achieves higher matching numbers and more correct matches in low-texture scenes.



讲义4 | Matchformer

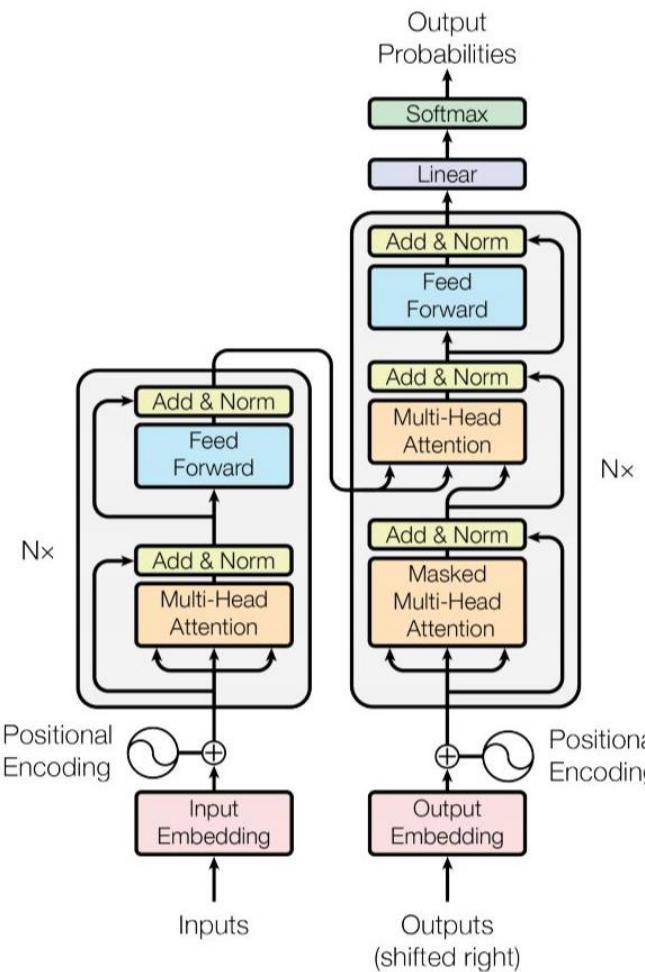
Matchformer Backbone

如下图所示，MatchFormer 采用层级transformer，通过四个stage来生成高分辨率粗略和低分辨率精细特征，用于局部特征匹配。在四个stage中，self-和crossattention被安排在一个interleaving strategy中。每个阶段由两个组件组成：一个positional patch embedding(PosPE) 模块和一组有效的注意力模块。然后，通过类似 FPN 的解码器融合多尺度特征。最后，通过 LoFTR 中介绍的粗略和精细特征来执行粗到精匹配。





讲义4 | Matchformer



Matchformer 模型使用了 **Self-/Cross-Attention**，为了帮助大家理解所以下面我们主要讲解 Self-Attention 机制。

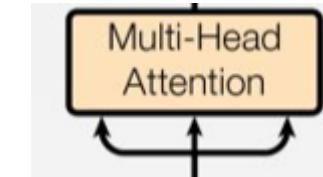


讲义4 | Matchformer

我们来介绍一下attention的具体计算方式。attention可以有很多种计算方式: 加性attention、点积attention, 还有带参数的计算方式。我着重介绍一下点积attention的公式, 这也是Transformer模型中所使用的attention公式:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

这里的Q K V 分别代表Transformer结构图中 (上一页) 橙色部分下面的三个箭头,





讲义4 | Matchformer

在我们之前的例子中并没有出现 Q K V 的字眼，因为其并不是公式中最本质的内容。

Q K V 究竟是什么？我们看下面的图

$$\begin{matrix} X & \times & W^Q & = & Q \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & \times & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \end{matrix}$$

$$\begin{matrix} X & \times & W^K & = & K \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & \times & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \end{matrix}$$

$$\begin{matrix} X & \times & W^V & = & V \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & \times & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \end{matrix}$$

知乎 @伟大是熬出来的

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

其实，许多文章中所谓的 Q K V 矩阵、查询向量之类的字眼，其来源是 X 与矩阵的乘积，本质上都是 X 的线性变换。

为什么不直接使用 X 而要对其进行线性变换？

当然是为了提升模型的拟合能力，矩阵 W 都是可以训练的，起到一个缓冲的效果。



讲义4 | Matchformer

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- 【一个小问题】为什么有缩放因子 $\frac{1}{\sqrt{d_k}}$?
 - 先一句话回答这个问题：缩放因子的作用是归一化。
 - 假设 Q, K 里的元素的均值为0，方差为1，那么 $A^T = Q^T K$ 中元素的均值为0，方差为d。当d变得很大时， A 中的元素的方差也会变得很大，如果 A 中的元素方差很大，那么 $\text{softmax}(A)$ 的分布会趋于陡峭(分布的方差大，分布集中在绝对值大的区域)。总结一下就是 $\text{softmax}(A)$ 的分布会和d有关。因此 A 中每一个元素乘上 $\frac{1}{\sqrt{d_k}}$ 后，方差又变为1。这使得 $\text{softmax}(A)$ 的分布“陡峭”程度与d解耦，从而使得训练过程中梯度值保持稳定。

讲到这里大家对self-attention有了一定了解啦，下面就要介绍 **Interleaving Self-/Cross-Attention**



讲义4 | Matchformer

Interleaving Self-/Cross-Attention

每个 **interleaving strategy** 都是self- and cross-attention modules的组合，其中每个stage包含 N 个注意模块，其中每个注意模块根据输入图像对表示为自注意或替代交叉注意。对于self-attention，Q和(K, V)来自同一个输入，所以self-attention负责图像本身的特征提取。对于交叉注意，(K', V') 来自图像对的另一个输入'。因此，交叉注意力学习了图像对的相似性。与提取匹配 LoFTR 在单尺度特征图上使用注意力并且仅在特征提取之后使用不同，

matchformer在编码器内部和多个特征尺度上交错自注意力和交叉注意力。

interleaving strategy能够同时学习特征并探索它们的相似性，即提取匹配方案。 由于浅层阶段的特征图强调纹理信息，所以在 MatchFormer 的早期阶段更多地应用了 self-attention 来专注于探索特征本身。深度阶段的特征图偏向于语义信息。

因此，发展更多的交叉注意力探索相似性更有效。

在注意力块内，提取自注意力特征，而特征对的相似性由交叉注意力定位。该策略更加人性化，它学习图像对的更多各自特征，同时关注它们的相似性。



讲义4 | Matchformer

Positional Patch Embedding (PosPE)

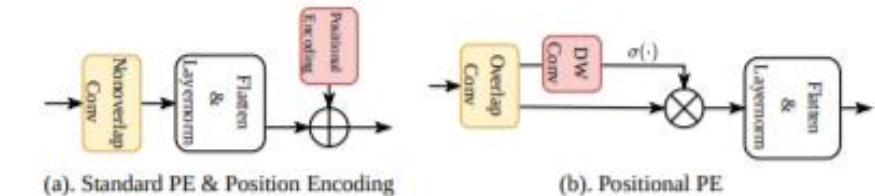
在经典的transformer中，将图像 ($H \times W \times 3$) 分割成大小为 $P \times P$ 的块，然后将这些块展平成大小为 $N \times C$ 的序列，其中 $N = HW/P^2$ 。

在这个过程中很难收集补丁周围的位置信息。结果，无法通过标准流程直接获取低级特征信息，这严重限制了局部特征匹配。

matchformer中提出positional patch embedding (PosPE)，用于捕获具有量参数的低特征信息，如右图所示。

它在第一阶段有一个 7×7 的卷积层 (padding 3 和 stride 2)，在后面的阶段有一个 3×3 的卷积层 (所有 padding 1 和 stride 2)，添加了一个深度方向的 3×3 卷积，以进一步增强局部特征并通过其填充操作对位置信息进行编码。

然后，在卷积的第一步之后，像素级权重由 sigmoid 函数 $\sigma(\cdot)$ 缩放。此外，matchformer 中 xiPosPE 包括第一个重叠卷积，用于捕获连续的patch区域信息。PosPE 增强了 patches 的位置信息并提取了更密集的特征，这有助于准确的特征匹配。





讲义4 | Matchformer

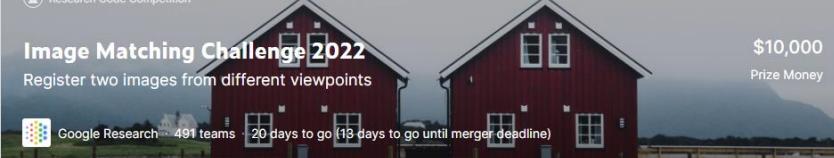
Efficient-Attention

经过Patch Embedding后，得到query Q、key K、value V，按照输入分辨率 $N=H\times W$ ，得到相同的 $N\times C$ 维度。传统注意力的计算公式为： $\text{softmax}((QKT)/\sqrt{d})V$ ，其中 \sqrt{d} 是比例因子。但是， QKT 的乘积引入了 $O(N^2)$ 复杂度，这在大图像分辨率中是禁止的，并且使模型效率低下。

为了解决这个问题，matchformer中使用两种有效注意力，即空间高效注意力（SEA）或线性注意力（LA）。然后复杂度由 $O(N^2)$ 减少到 $O(N^2/R)$ 或 $O(N)$ 。

因此，在特征匹配任务中使用纯基于转换器的编码器时，可以很好地处理和处理更大的输入特征图。

SEA降低了K和V的空间尺度。为简洁起见，将SEA作为self-attention描述为： $\text{softmax}((Q\cdot\text{SE}(KT)/\sqrt{d}))\cdot\text{SE}(V)$ ，其中 $\text{SE}(\cdot)$ 指先将序列 $N\times C$ 重塑为 $N/R\times(C\times R)$ ，然后将维度 $(C\times R)$ 投影到C。注意力模块的复杂度从 $O(N^2)$ 降低到 $O(N^2/R)$ ，其中R表示缩小比例。LA通过用核函数 $\text{sim}(Q; K) = f(Q)\cdot f(K)^T, f(\cdot) = \text{elu}(\cdot) + 1$ 替换原始注意力层的指数核来降低复杂度。利用矩阵乘积的结合性，可以先进行K和V的乘法运算。由于 $D\times N$ ($D \ll N$)，所以复杂度从 $O(N^2)$ 降低到 $O(N)$ 。

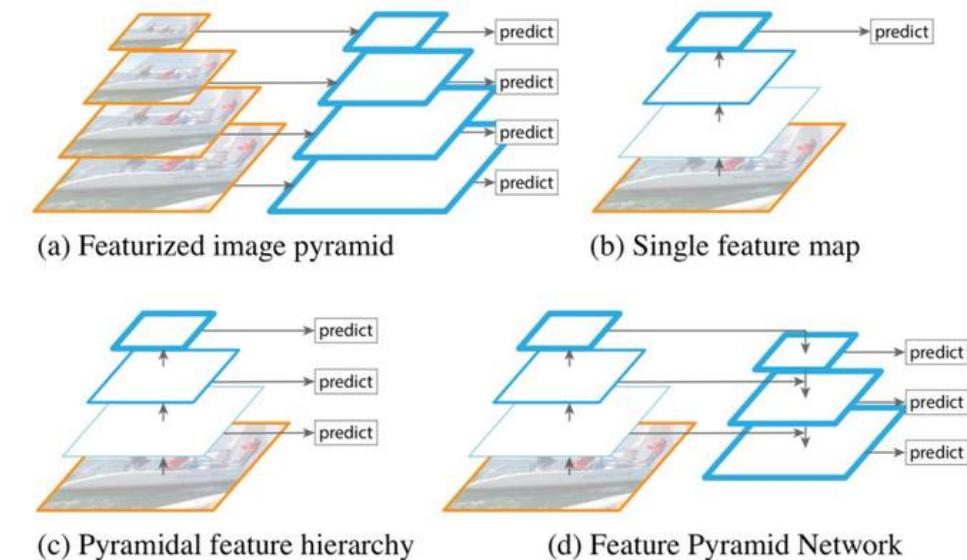


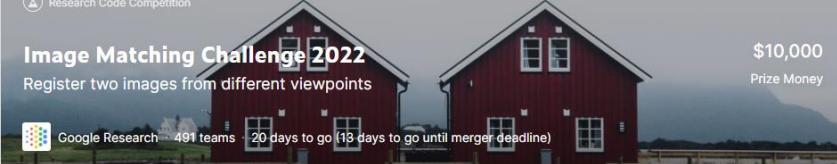
讲义4 | Matchformer

Multi-scale Feature Fusion

除了interleaving combination之外，matchformer编码器中有四个不同的阶段，其中特征分辨率逐渐缩小。与之前的仅考虑单尺度特征工作不同，MatchFormer融合了多尺度特征，生成稠密和匹配感知的特征进行特征匹配。matchformer的架构中灵活地采用了类似 FPN 的解码器，因为它可以带来两个好处：

- (1) 生成更鲁棒的粗略和精细特征以促进最终匹配；
- (2) 在不使整个模型计算复杂的情况下创建轻量级解码器。





讲义4 | Matchformer

Matchformer

Matchformer论文链接:

[MatchFormer: Interleaving Attention in Transformers for Feature Matching](#)

Matchformer源代码:

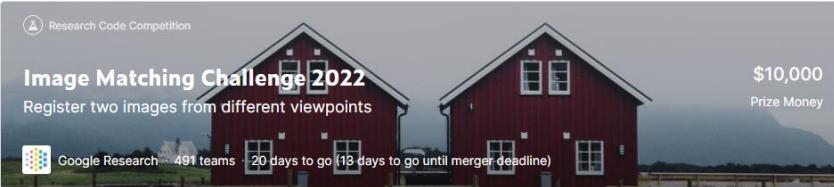
[MatchFormer: Interleaving Attention in Transformers for Feature Matching](#)



讲义4 | 模型融合

融合方法





讲义4 | 模型融合

Voting



Voting可以说是一种最为简单的模型融合方式。



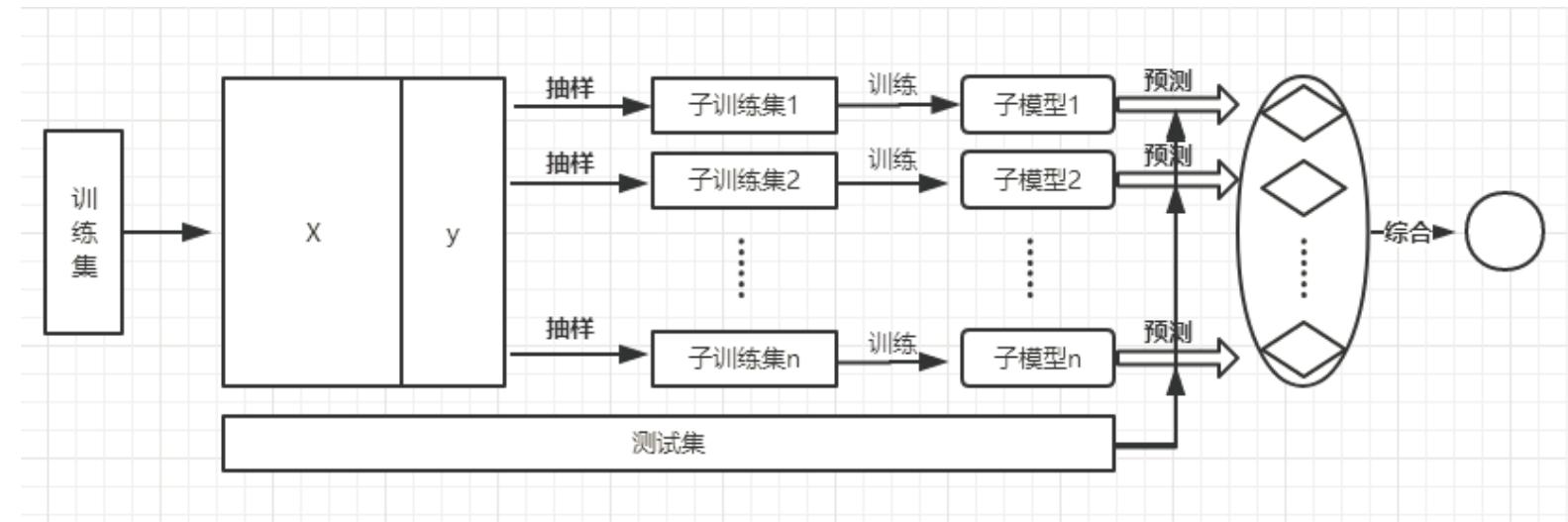
假如对于一个二分类模型，有3个基础模型，那么就采取投票的方式，投票多者为最终的分类。

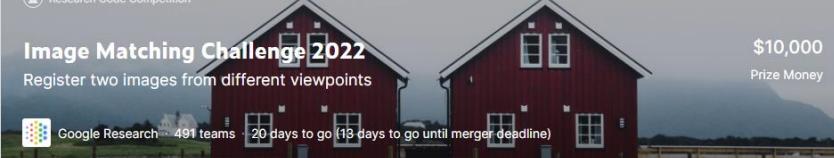
讲义4 | 模型融合

Bagging

Bagging的思想是利用抽样生成不同的训练集，进而训练不同的模型，将这些模型的输出结果综合（投票或平均的方式）得到最终的结果。

Bagging本质上是利用了模型的多样性，改善算法整体的效果。Bagging的重点在于不同训练集的生成，这里使用了一种名为Bootstrap的方法，即有放回的重复随机抽样，从而生成不同的数据集。



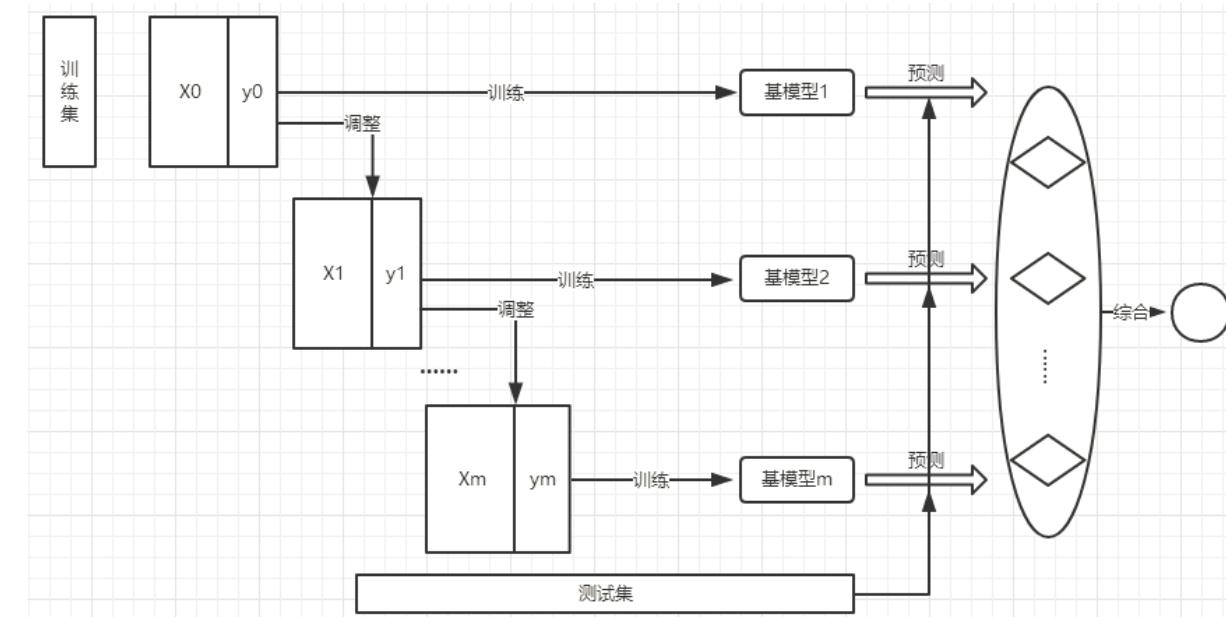


讲义4 | 模型融合

Boosting

Boosting是一种提升算法，其思想是在算法迭代过程中，每次迭代构建新的分类器，重点关注被之前分类器分类错误的样本，如此迭代，最终加权平均所有分类器的结果，从而提升分类精度。

Boosting与Bagging相比来说最大的区别就是Boosting是串行的，而Bagging中所有的分类器是可以同时生成的（分类器之间无关），而Boosting中则必须先生成第一个分类器，然后依次往后进行。核心思想是通过改变训练集进行有针对性的学习，通过每次更新迭代，增加错误样本的权重，减小正确样本的权重。知错就改，逐渐变好。典型应用为：Adaboost、GBDT和Xgboost。流程图如下所示：





讲义4 | 模型融合

Blending

类似概率voting

用不相交的数据训练不同的Base Model，将它们的输出（概率值）取加权平均。

	A	B	C	D	E
1		first	second	label	blend
2	1	0.8	0.2	1	0.38
3	2	0.6	0.7	1	0.67
4	3	0.2	0.3	0	0.27
5	4	0.3	0.1	0	0.16
6	5	0.9	0.7	1	0.76
7	6	0.1	0.3	0	0.24
8	7	0.8	0.8	1	0.8
9	8	0.3	0.6	1	0.51
10	9	0.1	0.2	0	0.17
11	10	0.6	0.4	0	0.46
12	loss	0.125	0.141		0.12276

=SUM((B2:B11-\$D\$2:\$D\$11)^2)/COUNT(\$A\$2:\$A\$11)}

=B2*0.3+C2*0.7

两列数据偏差越小，且方差越大，则blend越有效果



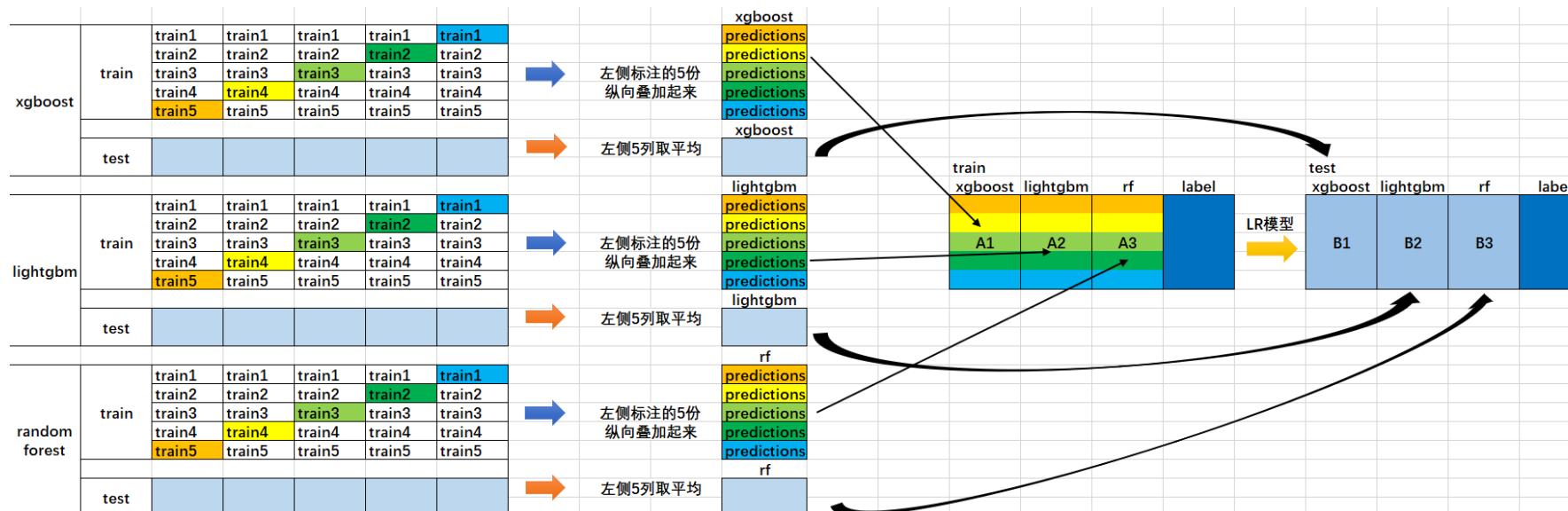
讲义4 | 模型融合

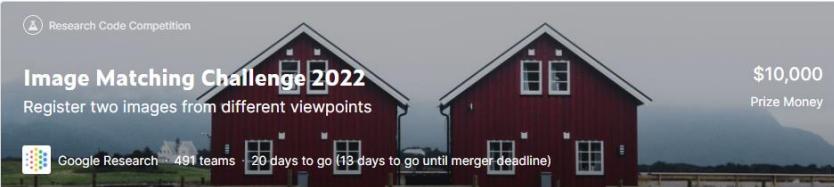
Stacking

交叉验证部分：首先将训练数据分为 5 份，接下来一共 5 个迭代，每次迭代时，将 4 份数据作为 Training Set 对每个 Base Model 进行训练，然后在剩下一份 Hold-out Set 上进行预测。同时也要将其在测试数据上的预测保存下来，对测试数据的全部做出预测。（**test预测矩阵1**）

5 个迭代都完成以后我们就获得了一个 **训练数据行数 x Base Model 数量** 的矩阵，这个矩阵接下来作为第二层的 Model 的训练数据，训练完以后，对 test data 做预测。（**test预测矩阵2**）

这时我们得到了两个预测矩阵，平均后就得到最后的输出。





讲义4 | 模型融合

本次比赛我们会采用 **Blending** 或者 类似 **Stacking** 的方法。

其中类似 **Stacking** 方法如下：

Train a simple regressor that takes you oof prediction vector and tries to predict the kaggle metric score (LB score) you will get submitting this kernel. no more words are going to be added about this!

Image Matching Challenge 2022

Register two images from different viewpoints

Google Research · 491 teams · 20 days to go (18 days to go until merger deadline)

\$10,000
Prize Money

文档



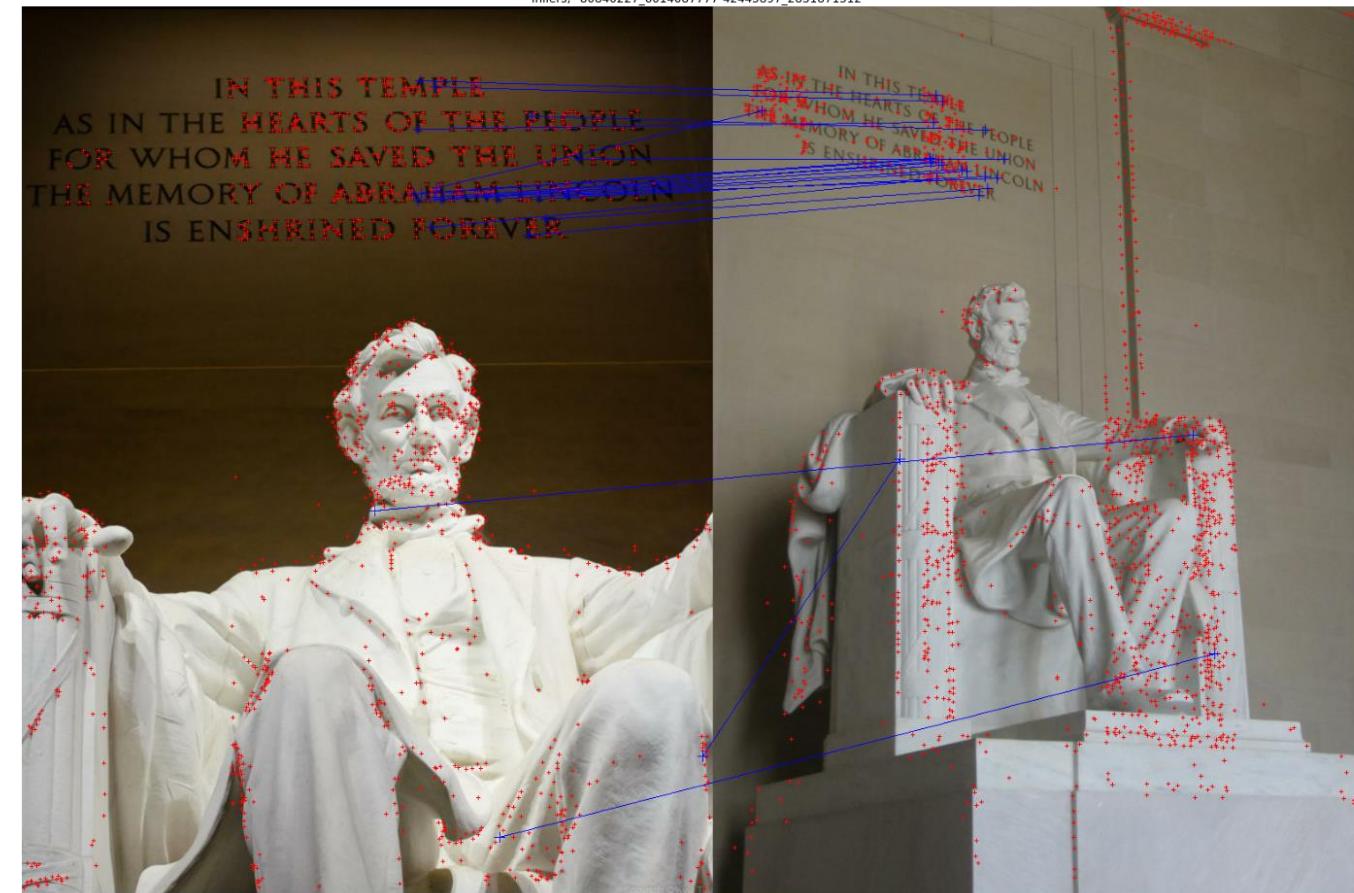
Image Matching Challenge 2022

Register two images from different viewpoints

\$10,000
Prize Money

Google Research 491 teams 20 days to go (13 days to go until merger deadline)

文档





比赛介绍

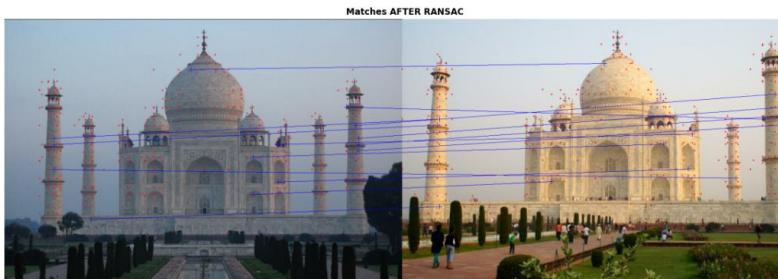
1 原图



2 关键点匹配



3 Ransac 或者 MAGSAC++ 过滤



留存高质量的匹配点将多余
低质性的关键点删除

4 计算Fundamental Matrix

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

用非常多的匹配点来拟合 最终确定九个值



比赛介绍

Data Explorer
2.65 GiB

< images (350 files)

- test_images
- train
 - brandenburg_gate
 - images
 - calibration.csv
 - pair_covisibility.csv
 - british_museum
 - buckingham_palace
 - colosseum_exterior
 - grand_place_brussels
 - lincoln_memorial_statue
 - notre_dame_front_facade
 - pantheon_exterior
 - piazza_san_marco
 - sacre_coeur
 - sagrada_familia
 - st_pauls_cathedral
 - st_peters_square
 - taj_mahal
 - temple_nara_japan
 - trevi_fountain
 - LICENSE.txt
 - scaling_factors.csv
 - sample_submission.csv
 - test.csv

Summary
5720 files
120 columns

Download All

This preview shows 30 out of 350 items. [Load more](#)

Data Explorer
2.65 GiB

< calibration.csv (118.9 kB)

Detail	Compact	Column	4 of 4 columns
A image_id	A camera_intrinsics	A rotation_matrix	A translation_vector
350 unique values	350 unique values	350 unique values	350 unique values
38521663_2805168218	9.379140e+02 0.000000e+00 5.245080e+02 0.000000e+00 9.378164e+02 2.850000e+02 0.000000e+00	9.494272e+01 1.000000e+02 -1.46982179e-02 9.99238024e-01 -3.63887227e-02 9.93638...	3.02136710e-02 4.60535159e-01 2.24622667e+00
76177997_2293073399	7.89452332e+02 0.000000e+00 5.320800e+02 0.000000e+00 7.89452332e+02 3.459800e+02 0.000000e+00	8.49184998e-01 4.39349712e-02 -5.26264722e-01 1.37128131e-01 9.4481788e-01 3.88663448e-01 5.0998324...	3.65419396e+00 -1.87177920e+00 -2.28747096e+00
28848887_5705734848	9.73220764e+02 0.000000e+00 3.750000e+02 0.000000e+00 9.73220764e+02 5.065000e+02 0.000000e+00	9.95465280e-01 -2.000000e+02 -1.91932112e-02 2.38499740e-02 9.99679886e-01 -8.44424145e-03 9.20879...	9.97120548e-02 -7.78547651e-01 -3.6359721e+00
48842381_9588988737	1.57201245e+03 0.000000e+00 5.140000e+02 0.000000e+00 1.57201245e+03 3.545000e+02 0.000000e+00	9.9898971e-01 2.95662099e-02 -3.61267693e-02 0.000000e+00 -3.52235951e-02 1.15701245e+03 -1.67663449e-01 3.06295...	-2.99884872e-02 4.89665524e-01 8.52235620e+00
23128878_9800439215	8.25811524e+02 0.000000e+00 5.015000e+02 0.000000e+00 8.25811524e+02 3.760000e+02 0.000000e+00	9.99424535e-01 2.34426347e-02 2.76675592e-02 0.000000e+00 -1.86993838e-02 9.8677678e-01 -1.16084539e-01 -3.10762...	-6.65801154e-01 -5.16997776e-02 -8.81406321e-01
21698821_985852958	8.37429564e+02 0.000000e+00 5.285000e+02 0.000000e+00 8.37429564e+02 2.830000e+02 0.000000e+00	9.95961155e-01 -7.00459617e-03 -9.9163322e-02 -1.000000e+00 8.7412594e+02 -1.8457215e-01 2.830000e+02 -1.9129735e-01 9.8520...	4.63463447e-01 8.12264599e-02 3.59848505e+00
59713287_6889584619	1.00124400e+03 0.000000e+00 5.180000e+02 0.000000e+00 1.00124400e+03 3.660000e+02 0.000000e+00	9.94835331e-01 1.70292345e-02 -1.07726763e-01 -2.10488270e-02 1.02124402e+03 9.99122549e-01 -3.62134768e-02 1.07869...	-1.68762763e-01 -5.47592738e-02 -4.8989878e-01
98157145_9159824129	6.41960120e+02 0.000000e+00 2.450000e+02 0.000000e+00 6.41960120e+02 2.450000e+02 0.000000e+00	9.96757980e-01 -8.04163799e-02 -2.10488270e-02 9.25894781e-01 3.78687882e-01 7.5375649...	-1.24214884e-01 -4.22131964e-01 -7.49487694e-01
82795354_283562282	1.16164005e+03 0.000000e+00 5.180000e+02 0.000000e+00 1.16164005e+03 3.425000e+02 0.000000e+00	9.35968848e-01 -7.29701867e-03 -3.54412379e-01 4.93154820e-02 9.92743745e-01 1.09971512e-01 3.510481...	2.84786422e+00 -8.01847364e-01 -1.97898324e+00

Data Explorer
2.65 GiB (3 directories)

< scaling_factors.csv (387 B)

Detail	Compact	Column
# scene	# scaling_factor	
16 unique values	16 total values	
british_museum	2.517	
brandenburg_gate	7.38	
buckingham_palace	18.75	
colosseum_exterior	36.99	
lincoln_memorial_statue		
notre_dame_front_facade		
pantheon_exterior		
piazza_san_marco		
sacre_coeur		
sagrada_familia		
grand_place_brussels		
st_pauls_cathedral		
st_peters_square		
taj_mahal		
temple_nara_japan		
trevi_fountain		
LICENSE.txt		
scaling_factors.csv		
sample_submission.csv		
test.csv		

Summary

- 5720 files
- 120 columns

Download All

No more data to show



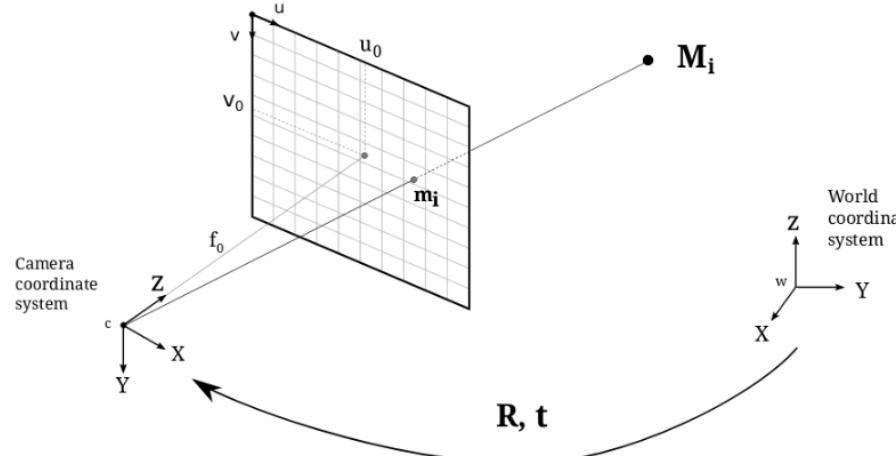
概念介绍

比赛的目标是估计两幅图像之间的相对姿态。这需要一些关于投影projective 和 对极几何epipolar geometry 的知识，特别是关于以下方面。

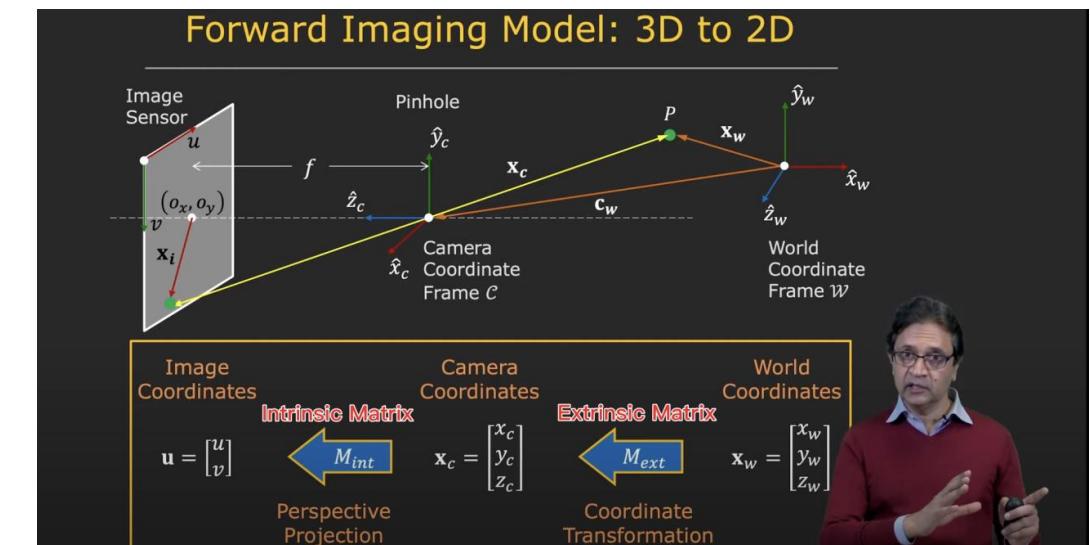
- **calibration matrix** 校准矩阵 \mathbf{K} 捕捉了决定三维点和二维（像素）坐标之间转换的相机属性。它也被称为相机内参 **camera intrinsics**。
- **rotation matrix** 旋转矩阵 \mathbf{R} 和 **translation vector** 平移矢量 \mathbf{T} 捕捉摄像机在全局参考框架中的 6 自由度姿势（位置和方向）。它们被统称为 相机外参 **camera extrinsic**。
- **fundamental matrix** 基本矩阵 \mathbf{F} 封装了同一场景的两个视图之间的投影几何。它不受场景中内容的影响，只取决于两台摄像机的内参和外参。

训练数据提供了 \mathbf{K} 、 \mathbf{R} 和 \mathbf{T} 作为 ground truth。参与者被要求估计 \mathbf{F} 。我们在下文中更详细地解释这些概念。

这里光栅看作相机的cmos

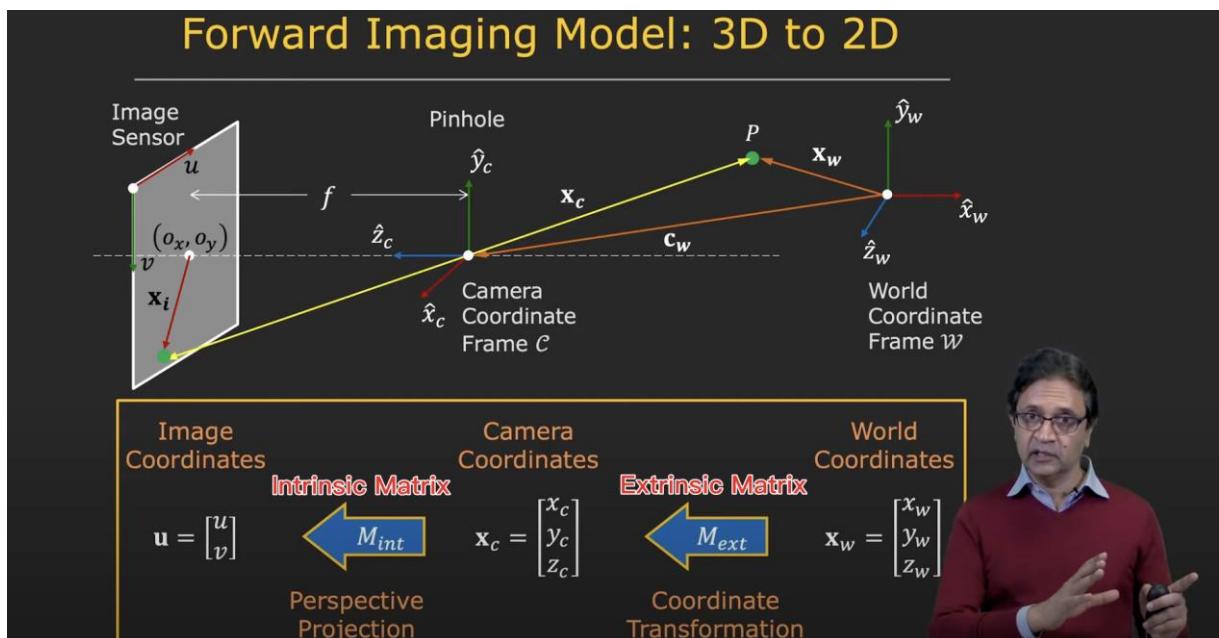


三维点 M_i ，以米为单位，投影到二维坐标 m_i ，以像素为单位，可以简单写成 $m_i = \mathbf{K}M_i$ ，其中 \mathbf{K} 为校准矩阵。





概念介绍



Intrinsic Matrix

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Extrinsic Matrix

Rewriting using homogenous coordinates:

$$\tilde{\mathbf{x}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$



概念介绍

Projection Matrix P

Camera to Pixel

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

World to Camera

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{u}} = M_{int} \tilde{\mathbf{x}}_c$$

$$\tilde{\mathbf{x}}_c = M_{ext} \tilde{\mathbf{x}}_w$$



概念介绍

Projection Matrix P

Camera to Pixel

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

World to Camera

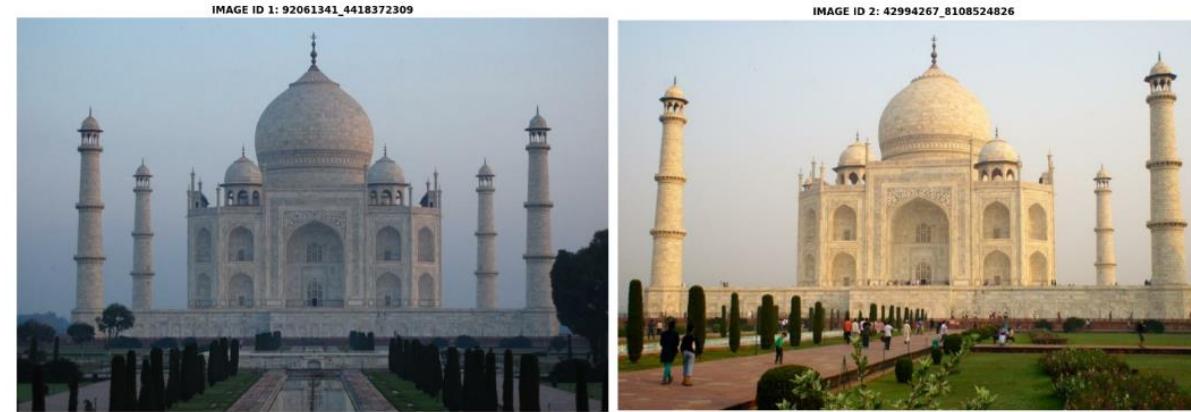
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{u}} = M_{int} \tilde{\mathbf{x}}_c$$

$$\tilde{\mathbf{x}}_c = M_{ext} \tilde{\mathbf{x}}_w$$



概念介绍



$$\begin{bmatrix} u_l & v_l & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

Fundamental Matrix F



概念介绍

$$\begin{bmatrix} u_l & v_l & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

Fundamental Matrix F

$$\begin{bmatrix} u_l^{(1)}u_r^{(1)} & u_l^{(1)}v_r^{(1)} & u_l^{(1)} & v_l^{(1)}u_r^{(1)} & v_l^{(1)}v_r^{(1)} & v_l^{(1)} & u_r^{(1)} & v_r^{(1)} & 1 \\ \vdots & \vdots \\ u_l^{(i)}u_r^{(i)} & u_l^{(i)}v_r^{(i)} & u_l^{(i)} & v_l^{(i)}u_r^{(i)} & v_l^{(i)}v_r^{(i)} & v_l^{(i)} & u_l^{(i)} & u_r^{(i)} & 1 \\ \vdots & \vdots \\ u_l^{(m)}u_r^{(m)} & u_l^{(m)}v_r^{(m)} & u_l^{(m)} & v_l^{(m)}u_r^{(m)} & v_l^{(m)}v_r^{(m)} & v_l^{(m)} & u_l^{(m)} & u_r^{(m)} & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A
(Known)

\mathbf{f}
(Unknown)



概念介绍

$$[u_l \ v_l \ 1] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

Fundamental Matrix F

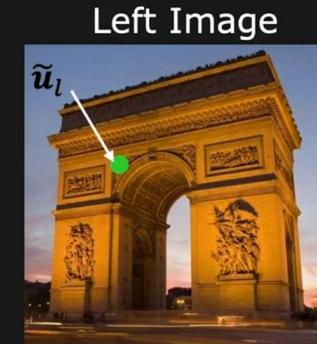
Finding Epipolar Lines: Example

Given the Fundamental matrix,

$$F = \begin{bmatrix} -0.003 & -0.028 & 13.19 \\ -0.003 & -0.008 & -29.2 \\ 2.97 & 56.38 & -9999 \end{bmatrix}$$

and the left image point

$$\tilde{u}_l = \begin{bmatrix} 343 \\ 221 \\ 1 \end{bmatrix}$$



The equation for the epipolar line in the right image is

$$.03u_r + .99v_r - 265 = 0$$



概念介绍

[Fundamental Matrix Demo \(cornell.edu\)](#)



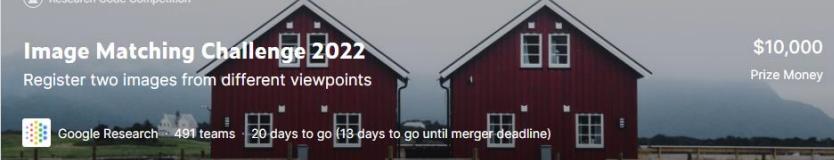
模型 和 库

- **SuperGlue 2019.11**
- [\[1911.11763\] SuperGlue: Learning Feature Matching with Graph Neural Networks \(arxiv.org\)](#)
- [CVPR论文笔记 | SuperGlue](#)

- **QuadTreeAttention 2022.01**
- [\[2201.02767\] QuadTree Attention for Vision Transformers \(arxiv.org\)](#)
- [https://github.com/tangshitao/quadtreeattention](#)

- **LoFTR 2021.04**
- [LoFTR: Detector-Free Local Feature Matching with Transformers](#)
- [https://github.com/zju3dv/LoFTR](#)

- **DKM 2022.02**
- [\[2202.00667\] Deep Kernelized Dense Geometric Matching \(arxiv.org\)](#)
- [https://github.com/Parskatt/DKM](#)



模型 和 库

SuperGlue: Learning Feature Matching with Graph Neural Networks

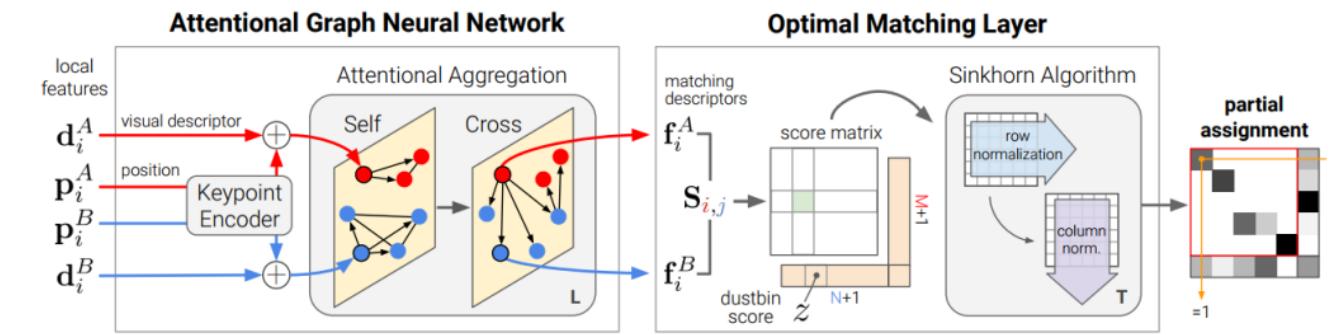
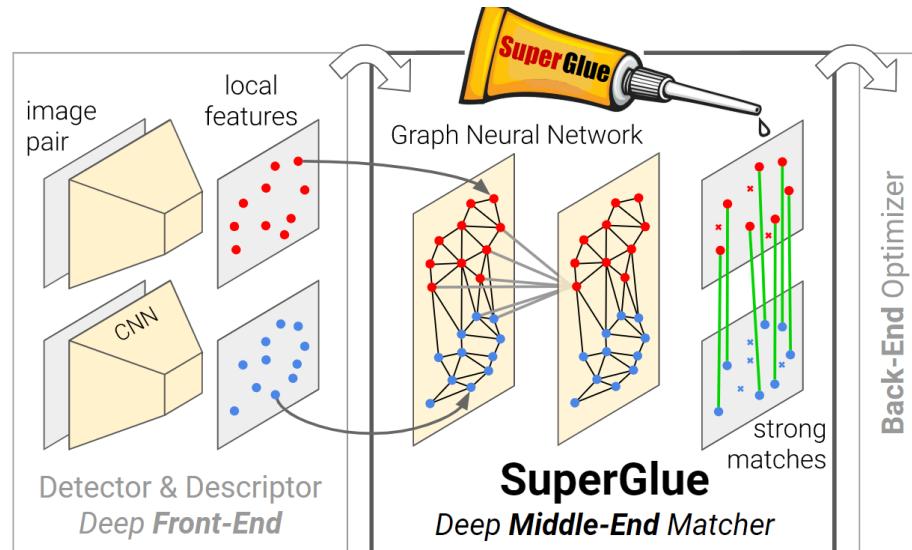
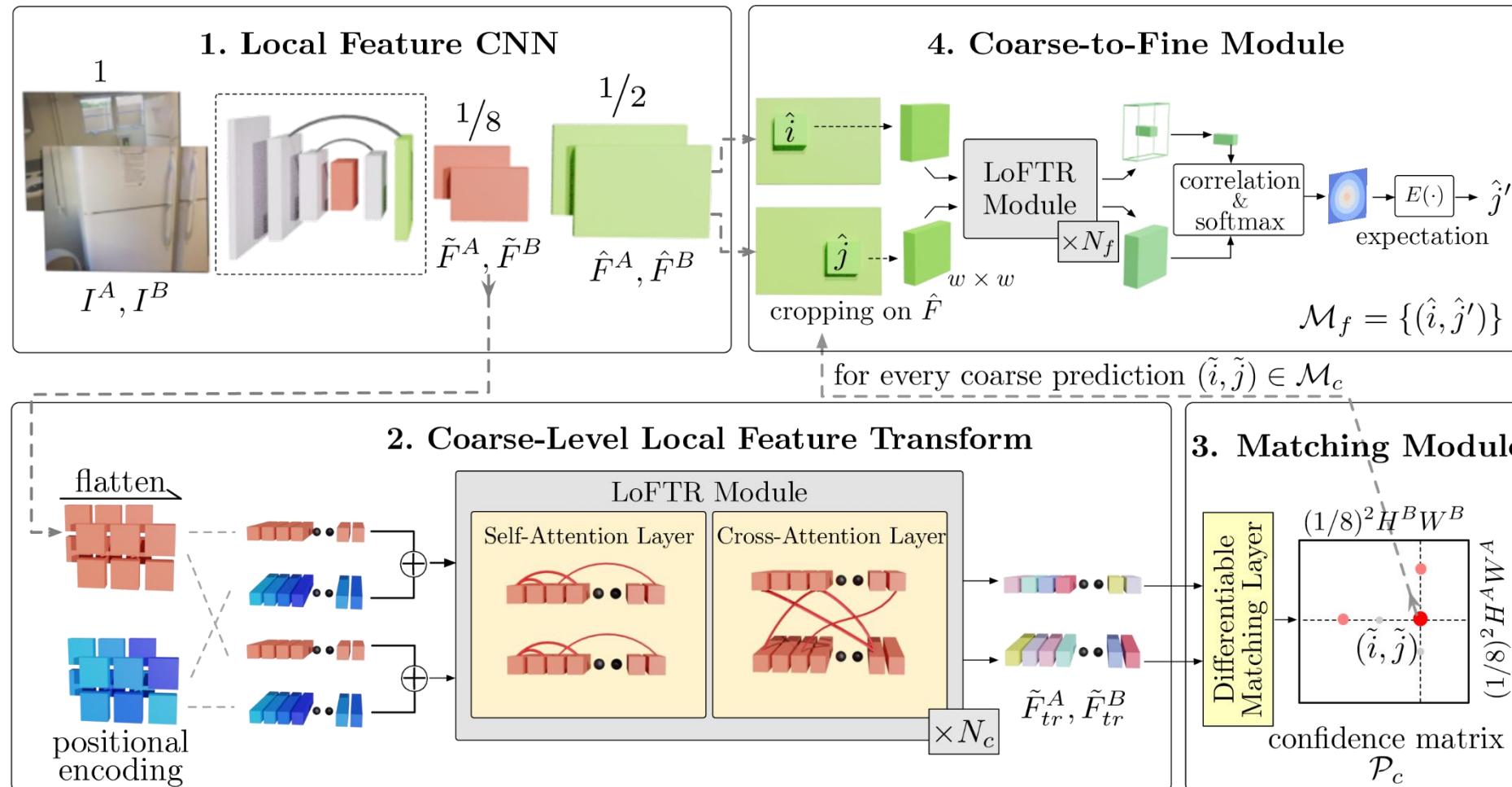


Figure 3: **The SuperGlue architecture.** SuperGlue is made up of two major components: the *attentional graph neural network* (Section 3.1), and the *optimal matching layer* (Section 3.2). The first component uses a *keypoint encoder* to map keypoint positions p and their visual descriptors d into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations f . The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations).



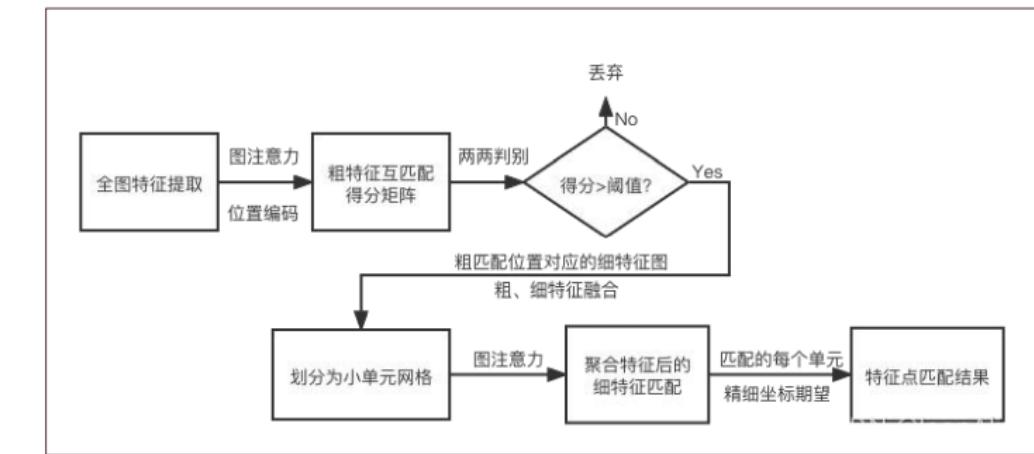
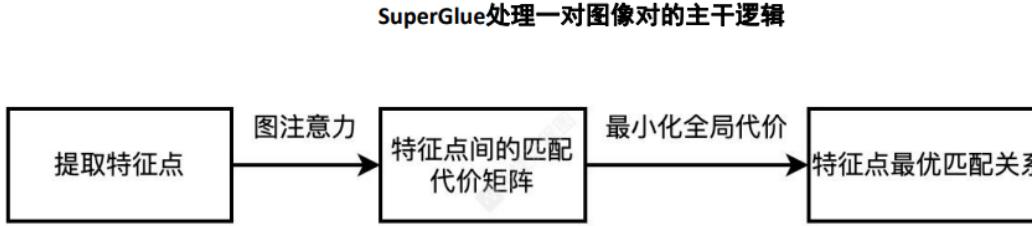
模型 和 库

LoFTR: Detector-Free Local Feature Matching with Transformers





模型 和 库





模型 和 库

QuadTree Attention

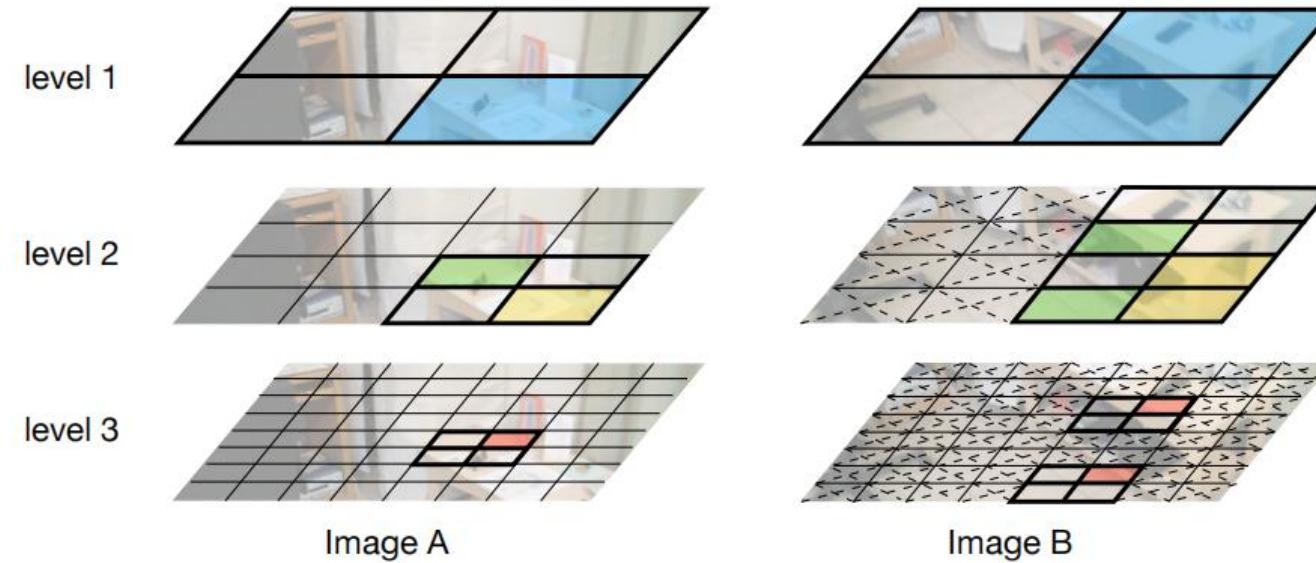


Figure 1: Illustration of QuadTree Attention. Quadtree attention first builds token pyramids by down-sampling the query, key and value. From coarse to fine, quadtree attention selects top K (here, $K = 2$) results with the highest attention scores at the coarse level. At the fine level, attention is only evaluated at regions corresponding to the top K patches at the previous level. **The query sub-patches in fine levels share the same top K key tokens and coarse level messages, e.g., green and yellow sub-patches at level 2 share the same messages from level 1.** We only show one patch in level 3 for simplicity.



模型 和 库

Deep Kernelized Dense Geometric Matching

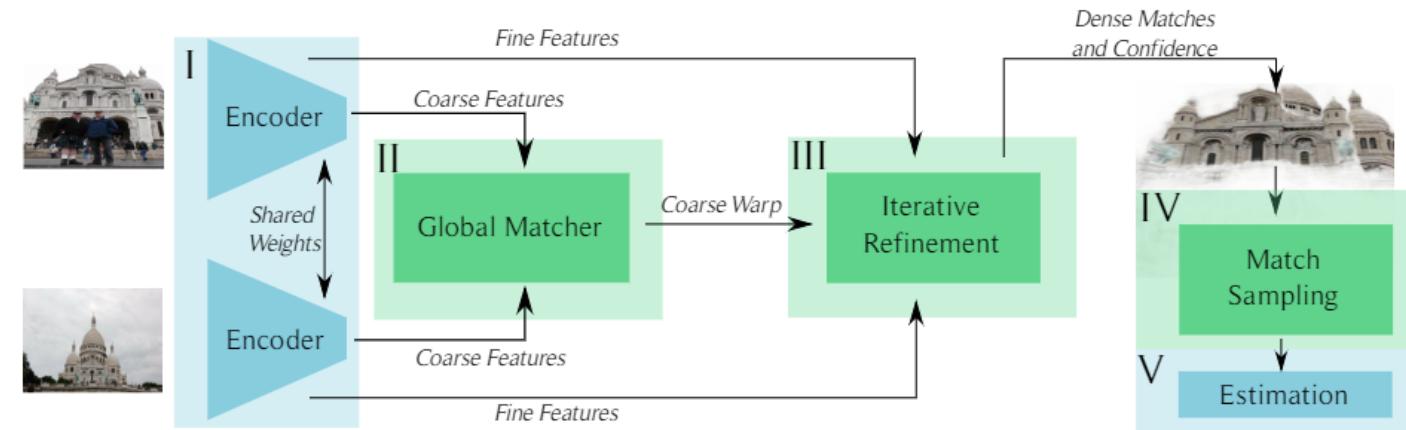


Figure 2: An overview of the geometry estimation by dense matching pipeline. **I:** In the first stage, a multiscale feature pyramid is extracted. We follow previous approaches and use ResNet-like encoders with shared weights. **II:** In the second stage coarse global matches are established. We improve this stage by viewing it as a probabilistic regression problem. We describe the regression problem and how we solve it in more detail in Section 3.2. **III:** The coarse warp is then refined. We propose a novel local-correlation free approach, which increases performance compared to previous approaches. This is detailed in Section 3.3. **IV:** Finally, for geometry estimation, a set of reliable matches need to be selected. We find that attenuation of the learned predicted confidence is beneficial. We further discuss this in Section 3.4. **V:** Once a set of matches have been selected, we use standard robust solvers for estimation as previous methods.



比赛介绍



TTA 数据增强
水平翻转



Thank you

