Fine-grained localisation

COMP90086 Final Project Report by Team 130

Zhaoxu Nie
The University of Melbourne
zhaoxun@student.unimelb.edu.au

Lihua Wang
The University of Melbourne
lihuwang@student.unimelb.edu.au

***Abstract*—In this project, we attempt to find a fine-grained location of images, based on other images that contain similar objects. More specifically, we examine methods based on feature matching and deep learning, to find out how each method perform on this task.**

Keywords—Feature extract and matching, Projection Geometry, Deep Learning

I. INTRODUCTION

Localizing the camera of an image becomes more and more popular in recent years. In this case, once we get an image, we could find matching in our database, then calculate the fine-grained location-based on them. Previously, the most popular to solve this problem is to use some feature detection and matching algorithms, but with the increase of computational power, deep learning-based methods become popular. In this project, we will examine different approaches, with a detailed evaluation.

The dataset contains 7500 training instances and 1200 test instances, while all of them are Google StreetView images taken at an art museum, including the indoor and outdoor environments. In the training set, each image has a location taken, while our task is to predict where the taken location of each test image.

II. METHODOLOGY

A. Images Matching

As the provided training dataset of images with position data came from the same museum environment as the test dataset, for simplification, we just assume the test images have the same position data as their closest match in the training image sets. To roughly obtain the estimation of the location of each test instance, the first step is matching each image in the test set to the most similar images in the training set via visual feature matching methods based on traditional feature extraction and deep learning techniques.

1) Traditional Feature-based Matching

Feature-based detection is the process of computing the abstraction of the image information, then making a local decision at every image point to see if there is an image feature existing at that point. It relies on the extraction of image features such, i.e. shapes, textures, colours, to match the target image or frame. Scale Invariant Feature Transform (SIFT) is a proven efficient feature detector applied in object recognition developed by Lowe in 2004 [9]. In the following sections, we compared four image matching techniques based on variants or extensions of SIFT.

a) Feature Detection with SIFT

We set the performance of the model that only use SIFT to detect features as the benchmark of our traditional feature-based matching approaches. Since SIFT is sensitively affected by light and only recognize objects by calculating the gradients, we first grayscale each image, then apply SIFT algorithm on them to extract features. After estimating scale-space extrema, we find key points which are localized and refined by eliminating the low contrast candidate points. Then, a key point orientation assignment based on local image gradient and lastly get the local image descriptor for each key point based on image gradient magnitude and orientation.

We performed FLANN matcher to match keypoints between two images by identifying their nearest neighbours since FLANN (Fast Library for Approximate Nearest Neighbors) algorithm is optimal for fast neighbour search in large datasets and high dimensional features. We apply the KNN algorithm to find the two nearest neighbours during matching. We consider a matching point as a good match if the ratio between the closest and second closest is greater than a given value. Now, for each test image, we got good match point sets of the whole training image. To measure the best matching training image, we calculated the ratios of good match points with total match points in two images. Considering a situation that two images were captured by the same camera location with different directions leading to a few scenes overlapping, the matching ratio might be low though they can accurately match some parts of the image. To address this problem, we apply the Homography matrix to avoid the rotating, transforming affection and find the best match image based on the partial matching.

b) pHash Preprocessing before SIFT

Since the time consumption of SIFT based feature extraction and matching is huge, we considered simplifying the time complexity by reducing each test image matching times with training images. We introduced an auxiliary similarity judgment method -- pHash (Perceptual hash algorithm) to roughly calculate the similarity of two images and sorted the top 500 most similar training images. Then apply feature detection based on SIFT technique simply repeating a) process.

pHash generates a string (hash value) that contains 64 binary values(0 or 1), for each picture based on DCT (Discrete Cosine Transform) computation and then compares the distance of the strings using Hamming distance. (The Hamming distance between the two equal-length strings is the number of different characters in the corresponding positions of the two strings). Smaller distance represents two more similar images.

c) ORB Preprocessing before SIFT

Another speed optimization method is replacing the pHash with ORB in b) process. According to our research, ORB is two orders of magnitude faster than SIFT without sacrificing the performance of feature detection as it keeps the features of rotation and scale invariance. So, we pre-rank the similarity of training images based on ORB feature matching ratios and sorted the most similar top 200 images to apply SIFT technique to match the best image.

2) Neural Network and Deep Learning

Besides the traditional targeting point matching methods, we consider the deep learning-based methods should also be

able to solve this problem. From a human point of view, when we are looking for a picture in a set of pictures that best matches the given one, we might expect the best match will have a very similar colour composition with the target, as well as should contain some common objects. Thus, we consider using CNN in this task, as it is quite powerful for object recognition.

Another consideration is the difficulties in training. We realize we just have 7500 instances, but 1500 unique labels, which will result in serious overfitting if we train a new model on this dataset. Thus, pre-trained models might be more suitable, because they have been trained in a very large dataset, and will not involve any overfitting issues on this downstream task.

More specifically, we are going to load a model, as well as load the pre-trained weights. After that, we will feed each train instance into the model, and take the output of the second-last layer as a result. The output will be a vector(the size of the vector will be varied based on different models), which can be considered as a low-dimensional representation of the original image. At the same time, we could conduct the same action on each test instance, then calculate the similarity of a test instance with each train instance, by calculating the dot product between two feature vectors(dot product is the cosine similarity). Finally, we assume that the taking location of the image we want to predict, will be the same as the image in the training set that has the highest similarity value with it. In other words, we use pre-trained models to do feature extraction, and compute the similarity for extracted feature vectors, then find the best match.

There are several pre-trained models off the shelf. To find suitable models, we discover the Imagenet Classification Benchmark [6]. We found the most state of the art models requires a large amount of computation power. Thus, considering our existing resources, we might not be able to load those complex models.

As a result, we decide to use a very simple model, VGG19, as our baseline. In addition, we attempt ResNet and produce a surprising result.

a) VGG19

To build a baseline classifier and verify the performance of deep learning-based models, we attempt a very basic model, VGG19, and set it as our baseline. VGG19 was invented in 2014, currently ranked 426 in Imagenet Classification Benchmark. It is quite simple to implement, as the Kears provide an implementation with pre-trained weights for it.

b) ResNet

The proposal of Deep Residual Network, ResNet, was a milestone in CNN-based image recognition. It uses a trick called residual learning, to reduce the degradation problem for traditional CNN, by adding the input directly to the output of some layers. If a network is too deep, which means it is composed of a large number of layers, the model might not be able to learn enough patterns from early layers due to the vanishing gradient problem. With the implementation of residual learning, the gradient for early layers will be preserved, and it could ensure the backpropagation algorithm could update the weights for those layers.

The Kears also provides several implementations and weights for ResNet, including ResNet50, ResNet101 and ResNet152. The main difference between these models is the number of parameters, so they will produce different sizes of output if we exclude the top classification layer. In addition, we believe once models have a similar structure, a larger size of output feature will be able to preserve more information, thus, we finally decide to use ResNet152 in our project.

In addition, our image size is (680,490,3), while the default input size of VGG19 and ResNet152 should be (224,224,3). However, as we exclude their top classification layers, there is no need to resize our images to the default size of those models, so the input of each model is just raw images.

B. Estimation of Camera Positions

a) Vote Results from SIFT, VGG and ResNet

We have plotted some results of the models, and find they tend to make mistakes on different images. Thus, we use an ensemble method taking a majority vote over the location predictions of the best performance in the 3 different models, ORB + SIFT, VGG19 and ResNet (We don't consider pure SIFT as a comparable model as it is not practical). To be specific, we vote the most common coordinates x, y respectively. In the situation that none of the three models has the same values, we calculate the average if the differences between three values do not exceed the threshold, otherwise, compare all possible pairwise combinations of the three and compute the mean of the minimum difference pair as the result. If the differences between the three models all exceed a threshold, we set the ResNet value as the final result.

b) Compute Locations via Projective Model

Based on the above image matching methods, we could match each test image with the top similar training images as well as their position data. As the parameters of camera view and images are known, we could compute the camera intrinsic matrix and apply the camera calibration algorithms and projection model (Equation 1) to refine the location of the test image.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} [R \mid T] \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

Equation 1. Projection model

The whole process was roughly shown below.

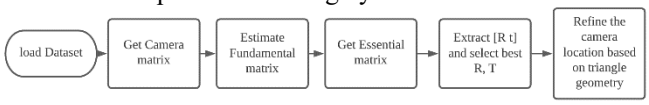

Figure 1. Calibration process

First, calculate the camera matrix (K). As the specification given, the camera has a 73.7 (FOV_h) deg horizontal x 53.1 (FOV_v) deg vertical field of view, and the size of the image is 680*490 pixels. We calculate the parameters of the camera matrix (K) based on the formulas.

$$f_x = W/2 * \tan(FOV_h/2)$$

$$f_y = H/2 * \tan(FOV_v/2)$$

$$c_x = W/2, c_y = H/2$$

To obtain the translation and rotation matrix which are extracted from the Essential matrix (E), we first calculate the Fundamental matrix (F) to solve for the E matrix. We use feature detection to find matching points (keypoints we obtained during the matching process) in the two images (test image and its matching training image), then solve for Fundamental matrix by simply applying the Opencv function *cv2.findFundamentalMat()* with Ransac.

Then, get the Essential matrix (E) from the Fundamental matrix using the following formula.

$$E = K^T * F * K$$

Next, decompose the Essential matrix (E) into R, t using SVD. There are four possible solutions of R, T.

$$t = U(0,0,1)^T = u_3, \qquad R = UW^TV^T$$

R	t
UW^TV^T	$+u_3$
UW^TV^T	$-u_3$
UWV^T	$+u_3$
UWV^T	$-u_3$

Table 1. Four choices for SVD solution of R, t

We need to determine which one is the right rotation and translation, at least the most suitable. The solution is finding the only one of the four configurations that all the points be in front of both cameras.

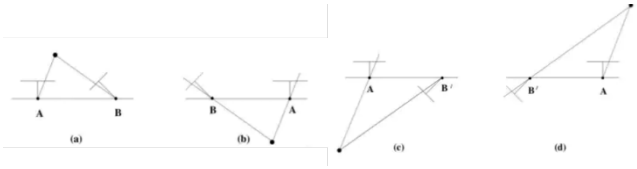

Figure 2. Four possible pairs of cameras

After obtaining the best [R|T], now we know the rotation and translation of the camera of the test image compared with the matching training image. However, since the position data from the training image were located in their coordinate system (view plane), to re-fine the test image camera, we need to locate the view plane first using two training image position data and these two images should have the view of the same object with the test image. Then we could calculate the test image position based on triangle geometry as the following picture shown.

There are two camera positions of training images $A(x_1, y_1)$, $B(x_2, y_2)$, we supposed camera of a test image is $C(x_3, y_3)$ and all of them locate the view of the same camera. The [R|T] for A to C we calculated as R_1, t_1, for B to C is R_2, t_2, the side length of B to C we set as t_3 by simply computing the AB length via $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Since we have known three side lengths as well as the two angles of a triangle, we could calculate the C coordinates based on triangle geometry.

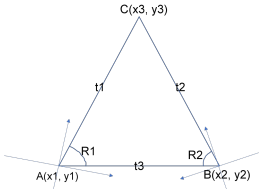

Figure 3. Triangle Geometry Diagram

III. EVALUATION AND ANALYSIS

Method	Performance on public test	Time Consumption
Pure SIFT	13.27	70h
PHash+SIFT	32.16	15h
ORB+SIFT	13.48	10h
VGG19	8.42	2h
ResNet152	5.42	12h
Voting	4.29	-

Table 2. Matching performance comparison

The table above shows the performance of public scores different models. Clearly, the deep learning model families outperform all traditional feature matching based models, and the voting classifier has the lowest error.

Experiments are conducted on our laptop, with 16Gb DDR3 memory, Intel Core 4720HQ and Nvidia GTX 980M. The deep learning-based methods are using GPU for calculation, while others just use CPU.

A. Traditional Feature-based

Comparedto the results of SIFT and its variants, it is noticeable that pure SIFT performs best on its image matching though it costs the longest time. After plotting the results compared with test images with their best match training image, we found that SIFT matched the keypoints well on most of the images even some part of the object is obscured. The reason is SIFT detects features only based on some local appearance points of interest on the object, no matter the size and rotation of the image changed. It also works well in the huge feature database to identify objects and seldom mismatch. However, it is time-consuming during the matching process due to repetitive operations and complex computation. For example, the Gaussian convolution kernel is used to convolve the image via multiplication and division operations which are resource-intensive in a computer. Overall, pure SIFT is hard to apply in practice due to real-time issues.

The feature detector combined with pHash and SIFT algorithms mismatched most with the shorter time than pure SIFT. Since we filtered training images by pHash and only keep the top 500 similar images for SIFT feature detection, the computation complexity was reduced (compared pure SIFT complexity 7500*1200 with pHash+SIFT 500*1200) which leads to obvious time consumption reduction. We supposed the reason for bad performance is inaccurate similarity comparison during pHash process. From pHash algorithm, we know that the image will be resized to 8*8 (64) pixels which might lose the highest frequencies and details of an image. Besides, to simplify the computation complexity, we grey-scaled the image and continuously reduced the DCT matrix causing the representation of hash value combination not enough. Bad performance from pHash affects the SIFT detection results which lead to the worst performance.

ORB detection contributes a more accurate similarity ranking for SIFT matching than pHash. It also computes faster than SIFT because it used FAST to speed up the extraction of keypoints and the BRIEF algorithm to calculate the descriptor. The unique binary string representation of the descriptor not only saves storage space, but also greatly shortens the computation time. However, we found that ORB performs not well to cope with scale transformation, for example, ORB cannot match well on images taken in the near and far view of the same object.

B. Deep Learning-based

The VGG19 model has a much better performance than those traditional models, while the ResNet152 have an additional remarkable improvement. On the other hand, the ResNet152 cost 6 times the time of the VGG19, because the extracted feature of ResNet152 has a larger dimension than it of VGG19, so computing the dot product between vectors requires a longer time.

In addition, we found that models that have a higher ranking on Imagenet Classification Benchmark, are very likely to have a better performance on this task. As a result,

we assume that models that ranked higher than ResNet152, might also have a similar or lower error.

C. Correction and Calculation

Vote

Vote result achieves the lowest error performance among all models above as it addresses some errors between multiple models to a degree. Based on our findings, incorrect matching results were varied and independent in each model, majority voting could achieve a better generalization effect by reducing bias and variance, that is, reducing the error under the premise of controlling the other. The reason why the error rate decreases exponentially is that, assume the error rates of our three models were almost 20%, to make ensemble result error, there should be at least 2 models having incorrect prediction, and the error probability is 20%*20% = 4%, which is much lower than individual model. Besides, we supposed another reason for better MAE is we calculate the average x, y values when three modes results vary. In a situation that the positions of the corresponding three images are closed, which means they might capture the same object from different views, by calculating the average values of x, y respectively, they might be closer to the real object location.

Camera location refinement

From the above methodology C(b), the results are not reasonable. We plotted 5 test images with some training images which have the closest coordinates, only 1 test image matches correctly. We supposed there must be something wrong at the step of calculating the 3D coordinates based on geometry solution. Though our theoretical derivation is feasible, there might be problems with the conversion and operations between matrices during implementation. For example, when calculating camera matrix (K), we got a negative value of focal length, however in real life, it is impossible for a camera has a concave lens, so we researched for the camera intrinsic parameters and found that a camera with given FOV parameters has a specific focal length as Figure 1 shown.

Lens Focal Length	35mm & FX format (1)		
	Horizontal	Vertical	Diagonal
24mm	73.7	53.1	84.1

Figure 4. The focal length parameters in real lens

D. Error Analysis

We plot and briefly go through some results to identify how our model made mistakes.

We found the SIFT model has a very bad performance when the test image contains a tree. In this case, it cannot find the corresponding tree in the training set, and they are very likely to find a wall as the best match. The reason we supposed that in Gaussian difference scale space, the leaves of a tree was processed into noise instead of feature key points so that they cannot be detected and extracted and most likely to be predicted as walls.

Figure 5. Mismatch on trees(middle and right are matched images)

However, for the VGG19 model, it can handle the tree well, we suppose the reason is the use of max pooling can handle some of the noises from trees. Besides, CNN can find capture the high-level structure of objects in an image, so it should be able to capture the shape and colour of a tree, then find the most similar tree in the training set.

On the other hand, the VGG19 are very likely to make mistakes on images containing grids, such as the window or wall (composed of rectangle stones). Just like the following figure shows, the VGG19 model can find an image that has similar grids, but it is not the real corresponding image.

Figure 6. Mismatch on windows

As for ResNet152, we do not find any patterns in its errors, and distinguishing whether it made an error prediction is even hard from human perception.

In addition, the test dataset has few images that can be considered as irreducible errors, each of them does not have any identifiable features, but only have walls or corners. The following image shows two examples of those irreducible errors. A possible way to find a matching of them is colour comparison, which means we could find a wall in the training set that has the most similar colour with them, but because we must find out those walls in test set manually and might involve a large amount of human work, we do not handle them in this project.

Figure 7. Irreducible errors

IV. FUTURE WORKS

A. Attempt Other Deep Learning Models

Since we found the deep learning models performs very well on this task, we are going to attempt further pre-trained CNN models in the future. According to Imagenet Classification Benchmark, there are a lot of models that have a better performance than ResNet152, and all of them might work well on this task. Currently, the CNN based model is not state of the art anymore but has overtaken those Transformer-based or attention-based models. As a result, we are passionate to attempt those models with attention mechanisms, if we have extra time and computing power.

B. Mixture of image

We found in the training set, there will be 5 pictures for each location, thus, it is reasonable that we could combine the 5 images, then treat them as a single image for matching. However, the detailed implementation should be considered carefully, such as the order of splicing, and what transformation should be done on each test image.

REFERENCES

[1] (2021). Retrieved 22 October 2021, from https://export.arxiv.org/pdf/1710.02726

[2] (2021). Retrieved 22 October 2021, from https://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/slides/CS4495-07-Calibration.pdf

[3] (2021). Retrieved 22 October 2021, from http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf

[4] FOV Tables: Field-of-view of lenses by focal length. (2021). Retrieved 22 October 2021, from https://www.nikonians.org/reviews/fov-tables

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2021). Deep Residual Learning for Image Recognition. Retrieved 22 October 2021, from https://arxiv.org/abs/1512.03385

[6] Papers with Code - ImageNet Benchmark (Image Classification). (2021). Retrieved 22 October 2021, from https://paperswithcode.com/sota/image-classification-on-imagenet

[7] photographs?, H. (2021). How can I calculate camera position by comparing two photographs?. Retrieved 22 October 2021, from https://stackoverflow.com/questions/10163034/how-can-i-calculate-camera-position-by-comparing-two-photographs

[8] Simonyan, K., & Zisserman, A. (2021). Very Deep Convolutional Networks for Large-Scale Image Recognition. Retrieved 22 October 2021, from https://arxiv.org/abs/1409.1556

[9] Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal Of Computer Vision, 60(2), 91-110. doi: 10.1023/b:visi.0000029664.99615.94