COMP90086 Computer Vision - Final Project

Fine-grained Localisation

Tuan Khoi Nguyen (1025294) Hoang Anh Huy Luu (1025379)

October 22, 2021

Introduction

In Computer Vision, image localisation is common for locating exactly where a picture is taken. This report will describe the group's systematic process of building the localisation model for 1200 images in the Getty Center in Los Angeles, USA, with the approaches containing a mixture between traditional feature extraction methods and new evaluation ideas, which resulted in some of our notable self-developed methods such as **Joint Adversarial Validation (JAV)**, **CumuLative Online Clustered K-Means (CLOCK)** and **Similar Histogram Online Clustered K-Means (SHOCK)**. These models have proven to be effective, and helped us in outperforming most competitors in The Kaggle Leaderboard [1].

1 Overview

With the output requiring coordinates in the form of `(x,y)`, this task can be seen in common problem types of geolocation or geotagging. This suggests solving directions of taking in spatial information such as object presence and color distribution, or neighboring information such as the location of a similar picture.

The Data

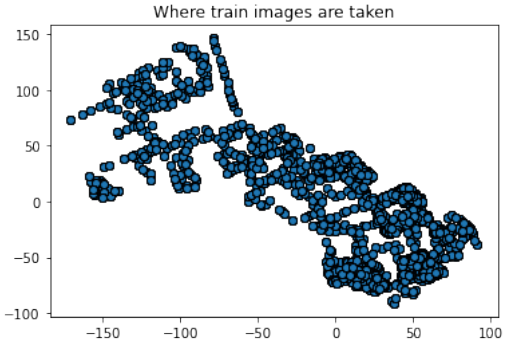

Figure 1: The location distribution of training dataset

Resulting plot from Figure 1 shows that the distribution for each data is diverse. While some locations have dense amount of images, some other areas have little to none images taken. This may result in conventional models not generalising well when a location has too few information known, or easily mistaken to another dense area with slight similarities. It is also noticeable that the data is only 2-dimensional, which may result in different pictures located within the same area.

2 Process Methods & Learner Models

The following section lists the learners used in this project. Given the task and data overview, the learners will focus on finding similarity criteria to match the test instances to their most similar train instance, and further test the approximation with translation algebra. It will also lists methods that aims in validating the data representation of the test set respective to train set, and getting the optimal set of hyperparameters.

2.1 Validation & Hyperparameter Tuning

Joint Adversarial Validation (JAV) To ensure accurate validation, assessment criteria for data representation is vital. Adversarial validation [1] measures similarity in distribution between 2 datasets, by attempting to distinguish them with a binary classification model. Lower accuracy means that the datasets have similar distribution, which makes each dataset being well-represented to the other. In order to pick the a validation set that will represent the testing dataset well, the candidate validation sets will have its train set representation compared against that of test set to get the most reflecting validation accuracy.

2-Layer Grid Search With large number of hyperparameters and limited time constraint, it is impractical to do a full grid search by going through each combination in small steps. Therefore, 2-Layer Grid Search, which selects combinations in longer steps to run the validation on, is chosen as the tuning method for this task.

The process will be done first on a large pool of combinations with large steps, picking the best combination, then narrow the search down to smaller steps, on surrounding candidates close to that combination.

2.2 Single Models

Except for the baselines, these models will all aim to assess similarity to pick the most similar images. Currently, the average location of these images will be taken.

Center Allocation, Random Allocation (Baseline) The former will assign the mean of all training locations to all test instances, while the latter allocate the locations randomly within training location range. These 2

[1] Ranked 3rd over 212 competing teams, as of October 22nd 2021

allocations methods rarely use dataset information, hence chosen as the minimum baseline for other models.

Convolutional Neural Network (CNN - Benchmark) CNN is the commonly used method for many computer vision tasks, which works by using pre-trained image datasets to extract image features. The model is chosen as the benchmark for its common use and well-known performance [2]. A number of different CNN models will be tested, each taking the final fully connected layer as the output to ensure that useful features are kept, while maintaining reasonable memory usage. The extraction result will measure similarity using Euclidean's Distance.

Scale Invariant Feature Transform (SIFT) SIFT works by matching similar patterns from color gradients, and count the number of matches [3]. For this task, SIFT will be used with Fast Library for Approximate Nearest Neighbors (FLANN) and filtered on Lowe's Ratio Test [4]. The similarity criteria is the number of matching points that passed the ratio test.

2.3 Ensemble Models

With the single models found and chosen, we attempted to improve on these models with combinations and modifications. Using model evaluation and domain knowledge, we have developed a number of extension or combination models for the localization problem.

CumuLative Online Clustered K-Means (CLOCK) Noting that the top candidates may include outliers, a dynamic clustering method is introduced. This method allocates the instances to clusters sequentially by getting one's distance to the centroids, and create a new cluster for every instace that is too far away. When having enough clusters or instances, the biggest cluster is chosen and its centroid will be determined as the final location.

CNN + SIFT (Combination) Noting that each singular model has their own weakness (which will be discussed after the Results section), we come up with the idea of sequential filtering. After filtering out weak images with CNN similarity, the model then rank the rest by SIFT matches, and use CLOCK to get the best average.

Similar Histogram Online Clustered K-Means (SHOCK) With the domain knowledge that all images are taken from the same camera, it can be assumed that images from the same area should share the same image colors. Therefore, color is chosen as an additional similarity criteria to filter out the outliers from CNN and SIFT. The number of histogram intersections is the similarity measurement, with larger number corresponding to more color-similar image.

It is noted that each similarity criteria has its own weakness that result in wrong prediction in small cases that can be omitted by other criteria, CLOCK clustering algorithm is expanded. Rather than choosing the biggest cluster, the model will vote between the biggest cluster, the cluster with the best SIFT matches, and the cluster with the most color matches.

Transform Calculation Now with the approximated location and the surrounding image cluster calculated for each image through CLOCK, and the domain knowledge that all images are taken with the same camera, the exact location can now be calculated using SIFT matching points and epipolar geometry. Further mathematical details will be described in Appendix A.

3 Setting Up

Operating Environment & Adaption The training process will be done concurrently across 3 devices to cut down running time and memory storage. Furthermore, to adapt to similar environments that lack resources, our implementation are built with the following attributes:

- Ability to run on any data partition rather than requiring the whole dataset. This way, training process can be divided across multiple devices.
- The process will export data segments to hard drive, and read them in again when needed, rather than storing them in memory. This way, the memory constraint is solved, and the procedure can always resume if there is a disconnection halfway.

Data Engineering & Storage Given that the size for each picture is large, we proceed to extract the essential features in a variety of ways to give our models good head start. We were able to extract features in 2 ways: using pre-trained CNN models, or SIFT matching. The images will then have their extracted features compared against training instances, then ranked and store to `.csv` files.

Validation While Kaggle public data contains 50% of the dataset, we were not able to fully evaluate model performance due to the submission limit, hence needed another validation method to rely. Given time constraint, we can only run validation process on partial data. Therefore, JAV is used as the assessing criteria, so that the chosen partition's experimentation results can be as reflective as possible for test set. The best data subset of 1/12 Hold-out (600 instances) will then be preserved for validation.

Hyperparameters Setting hyperparameters is done for CLOCK and SIFT, which have short running duration, has many parameters and is the base for most models. To save up the tuning process, a portion of parameters are either set to default, or chosen based on domain knowledge, namely as follows:

- SIFT with FLANN matching: The parameters are based on the recommendation of Lowe's Ratio Test [4]: 2 Nearest Neighbors and Ratio bound of 0.7.

- Epipolar transforming: The minimum number of SIFT matches should be 8, to satisfy the 8-point algorithm condition [5].

Running 2-Layer Grid Search on the rest of parameters showing that thresholding CNN feature distance within 5 units, ensuring cluster radius less than 3, and limit of up to 3 clusters brings the most optimal solution.

Model Execution After setting up and choose suitable tuning parameters, each model will be run sequentially, with top-performing ones of each stage have their result submitted to Kaggle to estimate generalisation. Feature extraction models will be executed first to get both similarity-based predictions and information to pass on the ensembles that will be run next. After the best areas are presumably found from the ensembles, epipolar geometry is then used to attempt on estimating more precise position.

4 Result, Experiments & Discussion

4.1 Baselines & CNN Feature Extraction

For all CNN models, all models will have the average location of top 3 best matching image as the final location.

Table 1: Result of baseline and CNN models

Model	MAE
Random Allocation (Baseline)	58.65
Centre Allocation (Baseline)	43.99
VGG19	29.24
ResNet50	9.82/**9.11**
ResNet101	9.74/**8.87**
ResNet152	9.90

0: 1/12 Hold-out Validation **0**: Kaggle Submission

The result shows that the CNN models outperformed the baselines well as expected. Out of 4 candidates, ResNet101 were found to be the best performer, and therefore was chosen for 1st Kaggle submission and processing in further models.

However, these CNN models have downfalls. Given that they base their similarity on features presence, which are likely present across multiples areas, the matching results can be mixed in with far away images that may cause wrong estimation.

Figure 2: Faulty CNN case example

Figure 2 shows an example of the problem, where 2 pictures from different locations are considered to be similar, as they both have trees and cement ground in the scene. This proves that a more precise matching intervention such as SIFT is needed.

4.2 SIFT

The running process shows that SIFT will not work for all instances, hence cannot produce complete output. This is because for images with no edges, SIFT will not be able to find any matching points to compare.

Figure 3: An image with no SIFT matching points

Error analysis also shows another problem. With some images containing noises, SIFT may mistake these noises as edge features and match them to the test instances, as shown in Figure 4. Therefore, SIFT cannot run on it's own, and will need imputation or combination with a different model to eliminate these noisy images first.

Figure 4: Case example of an image found to be best matched with different noisy wall images

4.3 Ensembles

Table 2: Result table of ensemble models

Model	Estimation Method	MAE
CNN	Top 3 Average	9.74/**8.87**
CNN	CLOCK	7.96/**7.67**
CNN + SIFT	Top 1 Average	4.59
CNN + SIFT	CLOCK	4.22
CNN + SIFT	SHOCK	3.86/**3.90**
CNN + SIFT	SHOCK + Epipolar	4.12/**4.06**

Chosen CNN Model: ResNet101

0: 1/12 Hold-out Validation **0**: Kaggle Submission

CLOCK & Combination Model As expected, CLOCK worked well in eliminating outlier estimations, and improved the outcome for both CNN & combination model. The combination model also proven to perform better than single CNN model, after crossing out irrelevant locations with SIFT matches. However, relying fully on identical features also have downfalls in some cases, especially when different viewing angles can return different textures or panorama stitches, resulting in detail loss.

Figure 5: Different orientations returning different details

Figure 5 shows an example of the case. Not only the window frame details are not in shape due to the panorama stitch, when looking at closer view on the left, new grid texture started to appear on side window due to glass reflection, making the picture being less recognizable than the other picture. This is when the color similarity comes in handy, where it can detect the similar color tones through histogram intersections, given the domain knowledge that the tone within an area should be identical.

SHOCK After taking color tone into consideration, the error is slightly reduced. This model is now presumed to produce the most precise approximated location for the test set, with mean error of 3.9 on Kaggle public dataset. However, some problems are still remaining.

Figure 6: Identical textures in different locations

Error analysis shows that like Figure 6, there are different locations sharing the same textures, which is common in wall images. Even human perception can hardly distinguish between these textures. Potential solutions to this problem may include high-level edge detection to find smaller visible details, or even using lighting geometry and image brightness to predict the taken location. However, with the time and resource constraint, these methods becomes infeasible, and can only be considered in the future where the conditions are more ready.

Figure 7: Identical outdoor views that are far away

Another problem that is common in outdoor view, shown in Figure 7 is that while 2 images can share the same details, they might be actually far away, given that the camera center can point to infinity. Therefore, epipolar geometry is believed to solve this problem, by estimating the translation vector between images.

Epipolar Geometry Contrary to expectation, the result with epipolar geometry not only showed no improvement, but also slightly decrease the performance. This shows that the outdoor view problem is not actually common in this dataset. Furthermore, with the current best mean error of 3.9, the algorithm is likely to have uncertainty that exceeded this value, making the estimation useless. Regardless of this, the algorithm will still become useful when more outdoor images are put into the dataset.

4.4 Potential Improvements

While our current model performed quite well on the dataset, there are still potentals remaining to improve the task, which were not able to be implemented due to the limit on data, time and resources. These includes panorama stitching images at the same area to make features more visible as a whole, or make a dynamic algorithm that changes parameters based on whether image is identidied as outdoors or indoors. Validation could also be done more completely with cross-validation, and rerun multiple times to give a more stable estimation.

5 Conclusion & Directions

Overall, through experimentation and error analysis, we came up with the ensemble model SHOCK, which consist of similarity detection using CNN features, SIFT matches and color histograms, and processed with CLOCK clustering method. This model has proven to perform effictively for the dataset, taking us to the top percentile of the Kaggle leaderboard. There will be even more space for development when more time and data are available.

With the locations now detected for each image, possible developments may include using Simultaneous Localization and Mapping (SLAM) to construct 3D views for the Getty Center, or even use the newly predicted instances to fill in the panorama stitches, making the Google Maps Street View better for user experience.

References

[1] Jing Pan, Vincent Pham, Mohan Dorairaj, Huigang Chen, and Jeong-Yoon Lee. Adversarial validation approach to concept drift problem in user targeting automation systems at uber. *arXiv: Learning*, 2020.

[2] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. 8(1), March 2021.

[3] D.G. Lowe. Object recognition from local scale-invariant features. IEEE, 1999.

[4] David G. Lowe. Distinctive image features from scale-invariant keypoints. 60(2):91–110, November 2004.

[5] R.I. Hartley. In defense of the eight-point algorithm. 19(6):580–593, June 1997.

A Translation Method

A.1 Intrinsic Camera Matrix

With Field of View (FOV_x, FOV_y) and size of image in pixels (w, h) given, the intrinsic camera matrix can be found as follows, assuming pinhole camera model:

$$K = \begin{bmatrix} \frac{w}{2tan(0.5\times FOV_x)} & 0 & \frac{w}{2} \\ 0 & \frac{h}{2tan(0.5\times FOV_y)} & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

A.2 Translation Algebra

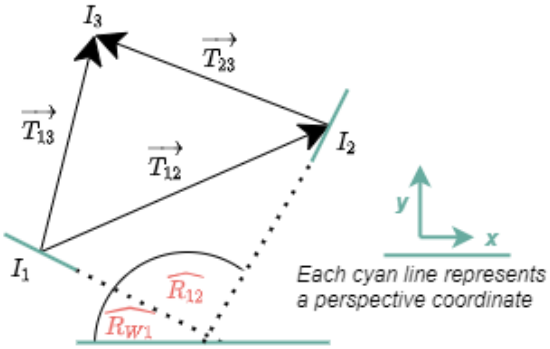

Figure 8: Arrangement diagram of epipolar geometry

This method requires at least 2 images I_1, I_2 with known coordinates to get the 3rd coordinate for image I_3, with each image having at least 8 point matches to perform the 8-Point Algorithm.

Assuming that the elevation is always the same. This way, any 2D real-world translation of (x, y) can be represented in 3D space as $T = \begin{bmatrix} x \\ 0 \\ y \end{bmatrix}$. For each pair of coordinates, the process can be done as follows:

- With K achieved, find the essential matrix E_{12} between I_1, I_2
- Decompose E_{12}, get rotation matrix R_{12} and translation matrix T_{12} to get from I_2 to I_1, in I_1 world coordinates.
- Get the angle of the translation vector T_{12}, in both I_1 coordinates and real-world coordinates W, using $arctan$ formula.
- Get the angle difference θ between 2 world coordinates, and get its rotation vector R_{W1}.
- Unit translation vectors to I_3 can now be calculated in real world coordinates:

$$T_{13} = R_{W1}T_{13(I1)}$$
$$T_{23} = R_{W1}R_{12}T_{23(I1)}$$

- Form vector equation: $x_1T_{13} + x_2T_{23} = T_{12}$.
- Solve $\begin{bmatrix} | & | & | \\ T_{13} & 0 & T_{23} \\ | & | & | \end{bmatrix} \begin{bmatrix} x_1 \\ 0 \\ x_2 \end{bmatrix} = T_{12}$
- Get image coordinate $I_3 = I_1 - x_1T_{13}$

For more than 2 known images, do this process for each unique pair, and put the results to CLOCK clustering to cancel out erroneous predictions.