Place Recognition at Union College

Qiyu 'Allen' Zhong

Thursday 17th March, 2022

Abstract

Through my studies at Union, I integrates my legacy of working in foundational courses of software development which tagging my . Throughout a few electives at Union College, such as Prof. Shane Cotter's Deep Learning and Prof. Nick Webb's Machine Learning, In this manner, I push myself towards re-collecting and organizing the dataset that I have been working around since the time when COVID-19 hit my college's campus in March of 2020. The instances in this dataset is mainly the one that I have been collecting through the campus with my phone. While being able to stay connected physically, the captures allow me to understand the buildings' distribution through the campus, and this experience inspired me to liaise with the Admissions Vice-President Matt Malatesta, from which I know that it is potential to enhance the UC Mobile software with a module that allows the user to take pictures through their visit while recognizing the building that is right in front of them. However, due to the limitation of time, I decided to focus mainly on putting a classification system that can handle the recognition task of the few most popular buildings.

Contents

1 Introduction

Therefore, I started to put up a machine learning system that attempts to recognize images, in which the buildings are included. The stuff that I am doing at this moment is mainly inspired by the course that I took at Peking University and Stanford University, during my first two years at undergraduate level. I am also interested in exploring the combinations of machine learning models and how they are performed on image dataset that are sparse and not really easy to deal with. In this manner, it would be more than grateful to be able to learn a way through handling the uncertainty of the dataset, along with the labelling of the sparse categories appeared throughout the time.

2 Background and Previous Work

Since my two Summer Sessions at Peking University and Stanford University, I was able to get along with the academic experiences in computer vision, where it can be applied back to social good, such as healthcare and environmental protection. The two sessions offered me a good context and solid toolset to get around with the researches in the field of computer vision, and in a good sense I am getting the track that I want to delve into by reflecting my interest in cars. Through the research seminar (CSC-497) as well as the first part of my thesis project (CSC-498), I found that I am really interested in the issue of segmentation in images, after exploring and analyzing a few autonomous-driving datasets throughout the data-processing libraries in Python. Therefore, I aim my self to work around with the dataset that I have been building since the early part of the COVID-19 pandemic. When a time comes for leveraging the situation, I wish that I would be able to gain a better understanding for the values that can come out from the analytics of the building dataset.

2.1 Place Recognition

Since the rising of using automation and artificial intelligence in the industry of tourism, the industry is trending with applications that actively work around with the custom place-recognition datasets, such as the one about street-level recognition[9], the exteriors that allows further feature extraction[5], etc. The scale is up to the research of autonomous vehicles; however, the dominance of convolutional neural networks (CNNs) upon the architecture potentially raises the concern of addressing recognition tasks with non-CNN models. Therefore, there have been work on finding a good feature extraction method with locally-aggregated descriptors[3], sensor-fusion through pre-processing[4], and prioritized feature-matching[6]. In the past eight years, since the large-scale issue of place-recognition opens for discussions and applications are published, it is noticed that the issue of having a relatively small dataset, which is what I am interested in tackling with. For preparing my work, I built up a dataset that contains the images and videos of, in which I took while I was walking through the campus and passing the exteriors of the buildings. In the For me, it is really an issue of knowing the deficiency of the data points, tackling the issue of having a relatively small dataset in the classification task. general, feel that it is related to the discussion of the theme , which relates to my college's campus as well as the theme that I am interested in, to further my interest in machine learning.

2.2 Histograms of Oriented Gradients (HOGs)

Mainly powered through projects that aim for detecting objects with bounding boxes, the sandwich-like architecture of this algorithm allows the machine to reveal the feature-like points in the image, with a unit of pixel. The first stage globally normalizes the instance so that the RGB values of each channel is computed with a square root, in which case the local shadowing would be addressed. Then, the gradient image is computed (in scikit-image, there is a parameter called orientations that defines the count of arrows that directs to the periphery of the pixel, in which the dominant color channel is selected. The following stage comes along with the computation of gradient, which basically groups a batch of pixels into a cell, and a line-up of cells into a block. This allows the data to reduce its dimensionality, in terms of only using the unit of cells to compute the gradients and their changes, stacked in corresponding bins.

In the first stage, the images are generated through the help of normalization. This is a stage where the R, G, B channels are mainly parsed as individual inputs that can be parsed upon the whole and current window. This gives us a less expensive input throughout the whole pre-processing stage, and it would be beneficial to the next stage where the colored images are processed and put into the computations of gradients.

In the second stage, the gradient-images are computed with a preprocessed layer of extracting the values for the locally dominant color channel. This helps to highlight the features that the images themselves have.

The stage before last integrates a normalization on the level of cells once again, along with a compression of vectors that is left off from the previous step so that the feature learning would be done. In terms of putting up the whole system, the Pipeline object is applied in the scikit-learn built-in package, as well as the original package of CNNs in the TensorFlow library is also applied throughout the machine learning pipelines.

I mainly applies the Histograms of Oriented Gradients (HOGs) feature learning algorithm to perform the stage of learning. It is also more than true that, given the completed form of the input image, we would be able to compute the gradients and the form of transitions through the rest of the experiments.[2] The essence of the algorithm breaks down the program into a few aspects: the gradients are split to different aspects based on the availability, and they can split to either a few directions of the mapping in this case. I would get the features such as cells_per_block, pixels_per_cell, and block_norm etc. throughout the whole piece of work, as they define the way that an image could be segmented throughout the pipeline and gotten processed. It means much to me to find a way of getting along with the raw image input, as they actually reveals the features in a landscape from multiple angles, bringing extra work for the

2.3 Scale-Invariant Feature Transform (SIFT)

The alternative descriptor from the SIFT localization algorithm also performs a powerful feature-matching algorithm throughout the time. The features such as the angles, the movement, the blockings, etc. are all bringing up the words of concern, to extract the edge-based features in the instances.

The big category of feature learning mainly focuses on finding the features provided by the localization module that attempts a better way of making the direction of changes happening in the area map. In the work by Li et al., the methodology of using SIFT as the main descriptor of an image dataset that contains buildings across the period of time is really impactful, at least on the leverage of the dataset's complexity and the remaining resources of the laboratory. I was really thinking that my algorithm only applies HOG would be potential to try out a different descriptor of the dataset, from which a comparative study could be performed. However, due to the inefficiency of timing in the middle-latter half of the term when I am concurrently handling applications, I was not able to devote my full energy in implementing a good experimental trial of the descriptor.

2.4 Feature Learning by Segmentation

In the most recent work around with the techniques for extracting the features from the significant points of the image, we have a recent discovery and validated work by the AI Lab at Stanford University, applying a few unsupervised learning algorithms that extracts the data points at the center of the images displayed on the screens.[8] It is also known to the field of place recognition, specifically a branch in computer vision, that

2.5 Few-Shot Learning

Since there is relatively a small size of data points that exists in my research dataset, it would also be insightful to address the progress of recognition task brought by performing a good output within a relatively small repository of the training dataset. In the field of machine learning, there is a branch called as few-shot learning, in which oriented and state-of-the-art research is carried out. Werthemier et al. [10] performed a research based on the complex realistic settings of the recognition tasks, and Bateni et al. addresses the B[1] improvement generated through the new architecture of the neural network that they get around with. In both papers, the curse of dimensionality becomes an issue that is tackled with multiple architectures, and the same applies to my project that there is relatively few count of instances that represents my images.

3 Methods and Design

3.1 Pre-processing: Dimensional Reduction

I worked on constructing a machine learning system that aims for recognizing the buildings inside the images that are included together in a scene of images. At this point, I was able to put up a manually-labelled dataset that includes 68 buildings across the campus, totalling up to 8504 data points. However, since there is only limited amount of RAM available through the running environment of Google Colaboratory, I attempted to run on recognizing all the included buildings but it turned out to be not as successful as we imagined. In the sense that getting around with the 12.96GB RAM allocated throughout the whole running environment. Therefore, I finally decided to get around with classifying the 4 most popular buildings based on the count of instances in the dataset. For this time, it enables me to put up a module that can pre-process the dataset, perform training and testing on scikit-learn-powered supervised-learning models for recognition, run data analysis on the results of the tasks, and validate the results with the help of confusion matrices, a type of evaluation metrics that is mostly handled through machine learning projects.

3.2 Pre-processing: Cross-Validation

3.3 Learning: Pipelining and Grid Search

3.3.1 Support Vector Machine

The support vector machine (SVM) model is noticed for being able to recognize discretized and sparse image input well, due to the principal-component-like computation on features.

The parallelled way of using the cores and the machine learning model itself creates the scenes of the data points themselves. The models generate the point where we would be able to get a hyperplane splitting the sides of the data points together, while not losing the other boundaries. Since the processing of sparse data such as the images of buildings would contain the signals where transition occurs in the most amount of time, making the model of SVM really stands out in segmenting the obvious features. The decision boundaries separate the data points where the lie at two sections of the images, bringing a bit further for the recognition task. Since the segmentation by HOG performs a good level of feature extraction, in terms of the transition between the edges of the buildings, support vector machine does a good job of separating the data points with a good hyperplane. ...

The parameters used in the built-in package of the classifier support vector machine includes the following few:

classify=SVC(C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)

3.3.2 Logistic Regression

Here we need to mention a concept called Generalized Linear Model (GLM). Since this family of models gets along with the parameters such as μ for the algebraic mean, a_0 for the initial term for penalization, and n for the space in the sample to start things around, fitting a version of logistic regression that is corresponding to the formula: $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{\theta^T x}}$. In the implementation of scikit-learn, we would be able to know that the stuff passed around is the version with extra arguments: This is actually generalized from the algorithmic family of GLM, setting the arguments that $\eta = \mu$, $T(y) = y$, $a(\eta) =$

It is more than famous for applying the logistic regression algorithm upon the usage of the complete dataset. The term $p(x) = 1/(1 + e^{-(x-\mu)s})$ is really telling the story that, given a normal distribution of mean μ and standard deviation s, it is more than direct to compute the value between the loss and the main branch itself. From a good level of tests, it would be direct to notice that things are doing pretty stable, no matter how we train the algorithm. So we would want to include the results of the computation for the probability.

classify=LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=1000, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)

3.3.3 Naïve Bayes

In Naïve Bayes, I applied the multiclass-labelling algorithm that deals with multiple inputs throughout the time, and it means much to take a further notice on the class to get together with, provided the following formula: $P(Y = x|X = x) =$

Naive Bayes classifier assumes that the features we use to predict the target are independent and do not affect each other.

But it is to validate that the work of Naive Bayes is actually performing better on the language and textual dataset, which means something totally different from our end.

Laplace-Smoothing for unlabelled instances to address the Zero-Frequency problem. This is applied in the field of NLP, but not really understood by what I have so far in the dataset.

There are even three kinds of Naïve Bayes algorithms: Multinomial, Bernoulli, and Gaussian algorithms. So far, my experiments apply the Gaussian Naïve Bayes throughout the whole process, based on the relatively normal distribution of my dataset. Though it would be necessary to admit that, given the distribution of classes in the dataset that I have at this moment, the Gaussian Naïve Bayes does not completely making sense. Instead, I should be trying the Multinomial Naïve Bayes because the features that I am using are not just simple.

Therefore, it would be critical to try different combinations of feature-extraction techniques and machine-learning algorithms.

classify=GaussianNB(priors=None, var_smoothing=1e-09)

3.3.4 Random Forest

The Random Forest (RF) model mainly involves aggregating simple and bimodal classification trees from a basic level, and they construct up to a few layers in the model of RF. For a good point of the model, it is treated and taken care as a model that generates

classify=RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0 bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)

3.3.5 k-Nearest Neighbors

The original assumption is the data exist in forms of clusters or exist in close proximity.

A good practice is to try and select an odd number for K to avoid equal numbers of votes and achieve a tiebreaker.

The support is the number of samples of the true response that lie in that class.

classify=KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

3.3.6 Linear Discriminant Analysis (LDA)

It is true that, LDA is used primarily an algorithm for reduce the dimensionality of the dataset, it is more than helpful to agree on the fact that the model

The LDA algorithm performs pretty well when it handles more than two classes in the task of classification. At this point, it means the robustness provided from a multiclass dataset, in which a logistic regression algorithm as a baseline might perform differently. From the view of mathematics, we would be direct to find that the discriminants, being the terms that is calculated as $n - K$, in which n is the number of instances in the dataset whereas K is the number of classes to classify. Since this term applies on the denominator of the each batch that in general the strategies and functions applies through; the other nice thing is that it wraps the implementation of Naïve Bayes during the stage of predictions, which is really corresponding to the type of problem that my thesis project is going to address. In this manner, it would be beneficial to take a different look on the computation of the complete formula: $\sigma^2 = 1/(n-K) * \sum((x-\mu)^2)$. The clustering context could be also applied here, but since our task deals with classification in the main branch, we would not need to consider further analysis.

Mathematically, LDA algorithm works under a normal distribution assumption of the dataset (a multi normal distribution assumption in fact). Note that LDA algorithm has shown a great robustness toward this criterion but will perform in a weaker manier if input data are significantly non-Gaussian. In a level of discussion upon my dataset, since I am mainly trying to compute the four classes that are least confused with each other, I attempted to gain a relatively equal distribution of the classes, but it turns out that most of the time the class of College Park Neighborhood mainly dominates the sample space of the dataset.[7]

If the current one integrates the precision that is larger than recall, that means the building is less likely to get confused with each other because its own feature stands out i.e. $TruePositive/(TruePositive + FalsePositive)$ gives the information about retrieved buildings. In the other way, if the current one integrates the precision that is smaller than recall, we would be able to know that the building is more likely to get confused with each other because its own features are confined with the ones from the other classes i.e. $TruePositive/(TruePositive + FalseNegative)$

classify=LinearDiscriminantAnalysis(solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None)

We assume that the data comes from a Gaussian distribution, i.e. where X as each instance comes from the distribution itself, with two classes k and l sorting out through the time of the application

$\delta_i(x) = \log f_i(x) + \log \pi_i$

$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log \pi_k$

Comparing to what I have gotten so far, it is more than true that the algorithm is stronger than what it may behave simpler than what would happen during the training process. In the training process of the algorithm, it turns out to be true that, tuning the hyperparameters comparing to the iors

3.3.7 Ada Boost

There is an obvious reason that Ada Boost fits the model in the most tacit and nice manner: the trees and functions set up as suppports from the basement level is really significant for getting things set along. In a manner that is fair enough: the trees are put up from the bottom forming an array of functions, which can build up to an ensemble model of items that gives further usage of training and recognition.

The Haar-like [1] features are really important in general for the lab to generate a good level of recognition. When the function is added up together like: $a_0 f_0(x) + a_1 f_1(x) + a_2 f_2(x) + ... + a_n f_n(x) + ...$, it would be computationally expensive in terms of the units of neurons added together in the model. However, using the model of reduction we would be able to cut off the weights of the model to be 5 or 6 features in the end of the day. Essentially, by computing the main and principle component in the function, it would be more than helpful to get around with the statistically significant features, i.e. the ones that are not attached with a minimal weight in the activation function. Therefore, it would be beneficial for us in general to think about the way of how Ada Boost can fit in an image classification task in general.

Regarding my own dataset, since the appearance and the edge of the buildings are really bringing further problems for the users to get along with, it would be very helpful to find an algorithm that can tune by itself during iterations, in terms of figuring out what are the most important features for the functionality.

classify=AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None)

3.4 Validation: Hypothetical Testing

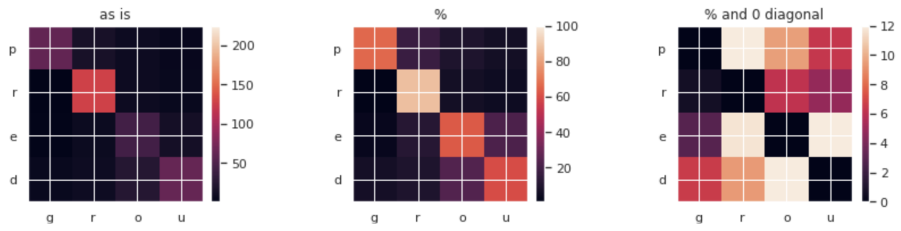
3.4.1 t-test: Results of Accuracies

https://www.machinelearningplus.com/statistics/t-test-students-understanding-the-math-and-how-it-works/

3.4.2 Evaluation Metrics in Experimental Context

Assuming that we are handling a good amount of evaluation metrics in a regression context, we would allow the way to apply the Mean Absolute Error (MAE), Mean Square Error (MSE), and Mean Absolute Percentage Error (MAPE). The project that I am handling here, however, is something that is about performing classification on the classes of buildings, which should mainly use the metrics of accuracy, precision, recall, and F1-Score. Additionally we use support to know the aggregated effects of the machine learning models.

The sample of images is attached below:

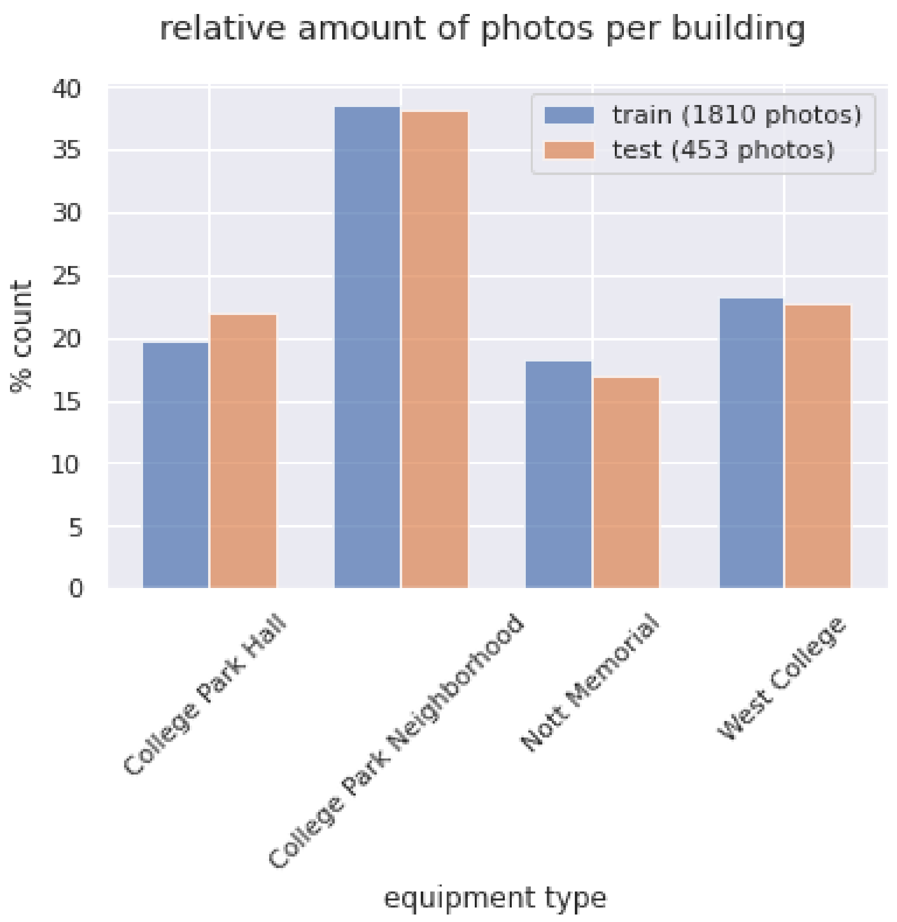


I mainly gained my analytics on the statistical reports as well as the metrics related with the whole project. So far, since I aligned for seven groups of input, I would agree that the reports are mainly about generating insights upon the confusion matrices and the classification reports. Taking the set of matrices from the angle of support vector machine, we infer from the leftmost matrix that, given the horizontal axis of ground truth and the vertical one being the signals, it would be convenient to know from the main diagonal about the number of instances that are correctly labelled. For the sake of normalization, the matrix in the middle of the graph highlights the percentage of correctly labelled instances. As showing the information of comparative study, it would be expected and beneficial to show the instances that are wrongly classified, so I included the third confusion matrix that reveals the information of confusion on the cells aside from the main axis.

In this case, I found that the model of supported vector machine has an overall accuracy of 76.3797% before optimization and 72.1550% after grid search. From the information of recall, I was able to sort out that it seems that College Park Hall and West College is less likely to get confused with the other buildings, since recall reveals the information about how the features of the other buildings are affecting the recognition of the current building.

3.5 Visualization: Bar Charts and Dash-Line Curves

The total amount of instances that are available throughout the project is distributed as the following manner:



4 Results and Discussion

4.1 Results

Besides, I was able to know that there are a few classes of instances that are not exactly labelled from the dataset in the images. In a good level of working around the input, I have been reading papers that draw insight about segmenting the contents in the instances, so that they become a bit clearer when the lines and bounding boxes are drawn. In my implementation at the stage of feature learning, it is good to know more about how the data is to be spread and distributed with a correct labelling process. Therefore, in my implementation, I aim for producing an evenly-distributed dataset for each of the classes that I am predicting, as well as finding a way of writing code for handling the complexity of the algorithms with the help of built-in and custom modules.

4.2 Discussion

5 Reflections

At this point, I really reflect that the Computer Science Thesis is the culmination of my CS education at Union College. The process began with the CSC497 seminar, in which I continued to develop the skills necessary for independent research, picking up the contents and techniques that I got around with my Sophomore Research Seminar (SRS). For this point, I agree that the Senior thesis serves as a way to document and present my capstone project. This is validated during my work in the term of Fall 2021, in which I piloted my project to put together the images. Typically, each term of your capstone experience includes an

Term	Course	Aims/Outcomes
Junior Spring	CSC497	Find Advisor, develop ideas
		A research question and fully formed proposal
Senior Fall	CSC498	Do lots of work and writing
Senior Winter	CSC499	More work, more writing
		A Completed Capstone Design Project

Table 1: A time line for a CS Thesis

intensive writing experience, resulting in a substantive final document. From the official website of Union College's Computer Science Department, I found this LaTeX template being helpful to serve for all the three written projects that I have been committed to since Spring 2021, with only modest changes to the template itself in the meantime.

This might be a good place for an example table. Table 1 shows a time line for a CS Thesis. Note that we are writing in **all** of the terms — we are not waiting until the end of the CSC499 to write up the thesis. Note: if you just use a tabular environment outside of a table environment, you still get a table, but it will be placed directly in the text where you put it, and it will not be shown in the List of Tables.

6 Future Work

All good things must come to an end. In this time, I allow myself to keep exploring the potentiality of the project, as it is not completely making sense to just focus on the classes that has the most count of instances on, but it makes sense to include the four classes that are least confused with each other, for the sake of easing the job that the system needs to handle, as well as enabling a version that can goes directly to the experimental software for further testing.

means much to me in terms of the Given the time of a compact term, I worked around to find the methodologies of cutting down the understanding the contents spread through the dataset.

(In this section you need to motivate your entire research project. What did you work on and why is it important?)

Appendices

A Terminologies

I apply the wording throughout the terms and concepts that I learned from the deep learning and machine learning courses at Union College's CSC-329 Deep Learning and CSC-321 Machine Learning. The collections of lecture notes and slides will be included when the draft is done.

B Acknowledgement

References

[1] Peyman Bateni et al. "Improved few-shot visual classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14493–14502.

[2] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[3] Jiadong Guo et al. “Local descriptor for robust place recognition using lidar intensity”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1470–1477.

[4] Stephen Hausler, Adam Jacobson, and Michael Milford. “Multi-Process Fusion: Visual Place Recognition Using Multiple Image Processing Methods”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1924–1931. DOI: `10.1109/LRA.2019.2898427`.

[5] Stephen Hausler et al. “Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14141–14152.

[6] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. “Location recognition using prioritized feature matching”. In: *European conference on computer vision*. Springer. 2010, pp. 791–804.

[7] Qiyi Lu and Xingye Qiao. “Sparse Fisher’s linear discriminant analysis for partially labeled data”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 11.1 (2018), pp. 17–31. DOI: `https://doi.org/10.1002/sam.11367`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.11367`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11367`.

[8] *Tutorial 3: Image Segmentation*. https://ai.stanford.edu/ syyeung/cvweb/tutorial3.html. Accessed: 15/03/2022. URL: `https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html`.

[9] Frederik Warburg et al. “Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2623–2632. DOI: `10.1109/CVPR42600.2020.00270`.

[10] Davis Wertheimer and Bharath Hariharan. “Few-shot learning with localization in realistic settings”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6558–6567.