Clustering Faces of Comic Characters

An Experimental Investigation

Artun Boz[1]

Supervisors: Lydia Chen[1]**, Zilong Zhao**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Artun Boz
Final project course: CSE3000 Research Project
Thesis committee: Lydia Chen, Zilong Zhao, Anna Lukina

Abstract

Face clustering is a subfield of computer vision and pattern recognition with many applications such as face recognition and surveillance. Accurate clustering of faces can also help us to create labeled datasets. However, in the domain of comics, face clustering is not well studied. Therefore, it is uncertain which methods of feature extraction and clustering perform well on faces of comic characters. In this paper, we investigate the effectiveness of comic face clustering. To conduct our investigation, we implement two pipelines: one that automatically extracts character faces from comic strips, and another that clusters the extracted faces. Using Dilbert Comics for our experiments, we examine the performance of various feature extraction and clustering methods. Additionally, we experiment with combining feature extraction methods and removing noisy samples to increase the clustering accuracy. We show that using color information is crucial for accurate clustering, and combining color with shape features further improves accuracy. However, our experiments indicate that accuracy improvement is not guaranteed for every combination of feature extraction methods. We also demonstrate that removing noisy samples using hierarchical clustering can increase clustering precision. Using our findings, we achieve an F1 score of 0.752 based on our Dilbert Comics dataset of 77,768 face images. We obtain this result by clustering 20,988 non-noisy face images into 35 clusters with a precision of 0.886.

1 Introduction

Face clustering is a well-established subfield of computer vision and pattern recognition. It has applications in domains such as facial recognition systems and surveillance [1]. The objective is to automatically group face images that belong to the same person based on visual similarities. Accurate face clustering can also help us create labeled datasets by collectively labeling images of the same cluster. In [2], the authors show the feasibility of this method by achieving an F1 score of 0.87 on the LFW dataset[1], which consists of 13K human faces.

The application of face clustering on characters from comics is similarly interesting. For example, images of comics with character labels can be used to train generative models for creative purposes [3]. However, the domain of comics presents unique challenges due to its exaggerated proportions and non-realistic color palettes. The authors of [4] show that prior techniques for face detection and recognition of real humans cannot be applied to characters from comics without a considerable loss of accuracy. Moreover, research on comic face clustering is not as developed as on clustering of real faces. It remains uncertain which methods of feature extraction and clustering are more effective for clustering comic faces.

In this paper, we investigate the effectiveness of face clustering applied to comic characters. We divide our investigation into three components. The first component is feature extraction. Having separable regions in the feature space is paramount to the success of clustering. We experiment with five methods of feature extraction to identify which works best. The second component of our investigation is to see whether we can improve the discriminative power of the feature extraction methods by combining them. Additionally, we experiment with autoencoding the combined features to improve clustering performance. The last component of our investigation is the choice of clustering method. We compare two types of clustering methods. These are the classical K-Means++ clustering [5] and an agglomerative hierarchical clustering algorithm called Approximate Rank-Order Clustering (AROC) [2]. The latter is particularly interesting, as it allows us to identify and remove noisy samples. Together, these components help us answer the following research questions.

- How effective are different feature extraction methods for comic face clustering?
- Can we improve the discriminative power of feature extraction methods by combining their feature vectors and autoencoding them?
- Can we outperform K-Means++ using Approximate Rank-Order Clustering and noisy sample detection?

2 Background

In this section, we first cover the state of research on comic face clustering. We then describe the various techniques we use in our methodology, including feature extraction, autoencoding, and clustering.

2.1 Comic Face Clustering

Face clustering, or more generally image clustering, can be divided into two sequential steps: feature extraction from images and the clustering of these features. Various feature extraction and clustering techniques have been studied for clustering human faces. For example, the authors of [6] use Local Binary Patterns (LBP) and Principal Component Analysis (PCA) to generate compact face representations for classification. In [2], the authors develop a variant of rank-order clustering that outperforms baseline techniques. With advances in deep learning, neural embeddings have also gained popularity. A prime example is FaceNet developed by Google [7], where the authors improve the representational efficiency of the embeddings while still achieving state-of-the-art face recognition performance.

Face clustering has also been studied in the domain of comics. In [8], the authors cluster manga[2] characters using the DBSCAN clustering algorithm based on features extracted using a convolutional neural network. In a newer paper, the same authors follow up with a character re-identification pipeline that leverages the "Face-Body and

[1] https://vis-www.cs.umass.edu/lfw/

[2] A style of Japanese comic books and graphic novels.

Spatial-Temporal Associated Clustering" method [9]. Simply put, the pipeline uses a classification loss on pseudo-labels and a contrastive loss on extracted features to train the feature extraction network and the clustering algorithm. Unfortunately, the authors do not share their code. Apart from these two contributions, research on clustering of comic faces is lacking compared to its counterpart in real-life faces.

However, it is worth mentioning the broader literature on feature extraction techniques applied to comic images. For example, the authors of [10] evaluate the effectiveness of various feature extraction techniques for cartoon pornography detection. Furthermore, the authors of [4] introduce features tailored for anime[3] characters. These features include skin-color regions and edges. The authors of [11] show that combining shape and color characteristics improves accuracy in object detection applied to cartoon images. Deep learning has also been studied in the domain of cartoons for tasks such as supervised face detection [12, 13].

2.2 Feature Extraction Techniques

Local Binary Patterns (LBP)

Local Binary Patterns (LBP) capture the local texture information of pixels in an image [14]. The method compares the intensity of each pixel with a circular set of surrounding pixels. The neighboring pixels that have a higher intensity are multiplied with a binary coefficient, and the resulting values are summed to obtain a pattern for the given pixel. The pixel patterns are used to construct a histogram that compactly represents the image. The method is particularly effective at identifying corners and edges.

Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) is a feature extraction method based on local gradient information [15]. The pixel intensity gradient is computed per pixel, the pixels are joined as cells, and a histogram of the gradient directions is computed for each cell. The contribution of each pixel to the histogram is scaled by the magnitude of its gradient. Finally, the histograms of neighboring cells are grouped as blocks and normalized. The concatenation of the histograms forms the feature vector.

Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) method has two components: FAST and BRIEF [16]. FAST is an efficient corner detector that relies on intensity variations around pixels to identify keypoints in an image [17]. Following keypoint detection, ORB computes descriptor vectors for each keypoint using a variation of BRIEF [18]. BRIEF generates a binary string that represents the local image patch around a keypoint. After obtaining a descriptor vector for each keypoint, a vector quantization method is applied to combine the vectors into a single descriptor vector per image. Examples of vector quantization methods include bag-of-visual words [19] and fisher vectors [20].

[3] A style of Japanese film and television animation.

SimCLR

SimCLR is a representational learning framework developed by Google Research [21]. The framework trains a deep convolutional neural network called the backbone with augmented training data using a contrastive loss function. The augmentation step generates two versions of a training image called positive pairs x_i and x_j. The goal is to have the backbone learn similar representations for x_i and x_j. After training, the backbone is detached and can be used to create latent representations of images for downstream tasks.

Autoencoding

Autoencoding is a method of learning representations of data points that have lower dimensions than the original data. The method consists of an encoder that maps an input to a latent representation and a decoder that reconstructs the input from the latent representation [22]. A classical example is Principal Component Analysis (PCA). Neural networks are also widely used for autoencoding. Deep autoencoders use multiple fully connected neural networks as the encoder and decoder.

2.3 Clustering Algorithms

K-Means++

K-Means++ is a variant of the K-Means clustering algorithm that improves speed and accuracy [5]. The difference lies in the selection of k initial centers of the clusters. Instead of being randomly selected at the same time, the centers are selected in an iterative fashion. At each iteration, data points further away from the current centers have a higher probability of being selected as a center. As a result, the centers are not placed very closely to each other. The authors show that K-Means++ provides a considerable improvement in the final error of K-Means clustering.

Approximate Rank-Order Clustering

Approximate Rank-Order Clustering is a type of agglomerative hierarchical clustering technique that uses a distance measure based on nearest neighbors [2]. The distance between two clusters is defined as the minimum distance between any two samples in the clusters. The distance between two samples a and b is given in Equation 1 and Equation 2.

$$D(a,b) = \frac{d(a,b) + d(b,a)}{\min(O_a(b), O_b(a))} \tag{1}$$

$$d(a,b) = \sum_{i=1}^{\min(O_a(b),k)} I_b(O_b(f_a(i)), k) \tag{2}$$

where k indicates the top-k neighbors, $f_a(i)$ is the i-th face in the neighbor list of a, $O_b(f_a(i))$ gives the rank of face $f_a(i)$ in face b's neighbor list, and $I_b(x,k)$ is an indicator function that equals 0 if face x is in face b's top k nearest neighbors, and 1 otherwise. In simple terms, the distance function gives lower values as the presence of shared neighbors increases. To mitigate the $O(n^2)$ computation of the nearest neighbors, the algorithm uses the randomized k-d tree algorithm to compute approximate nearest neighbors.

3 Methodology

Our investigation of comic face clustering revolves around feature extraction, feature combination, and clustering. Observing that these steps have a sequential dependency, we base our methodology and evaluation on a pipeline that facilities all of these steps. The input of this pipeline is images of comic character faces, and the output is cluster labels. The pipeline itself consists of the following steps: preprocessing, feature extraction, feature combination, autoencoding, and clustering. The pipeline is visualized in Figure 1.

Unlike images of celebrity faces, however, the images of comic character faces are not readily available in large quantities. Therefore, we also need to create a dataset of comic character faces, i.e. the input of our face clustering pipeline. To this end, we develop another pipeline that takes raw comic strips downloaded from the Internet as input and extracts character faces from the strips as output. The steps of this pipeline are the following: separation of the comic strips into panels, removal of text from the panels, face detection, and face extraction. Figure 2 contains a visualization of this pipeline for an example comic strip.

In the rest of this section, we describe our methodology for each pipeline step by step. The code used for both pipelines is made public on GitHub. Refer to section 6 for the link.

3.1 Face Extraction Pipeline

Panel Extraction and Text Removal

Most comic strips are made up of panels arranged in sequence. Before running a face detection algorithm, we separate the comic strips into distinct panels. Then, we remove text boxes from the panels to prevent the face detection algorithm from falsely identifying chunks of text as faces. For panel separation, we leverage a portion of the "Automated Text-Image Dataset Creation Pipeline" developed by [23] and keep the default hyperparameters. For text removal, we use "ImageAnnotatorClient.document_text_detection" method of Google Vision API. We run the detector to locate text blocks and remove those that have a confidence greater than 0.8.

Face Detection and Extraction

The next step of the pipeline is face detection and extraction. In our implementation, we use wrappers around the "Domain-Adaptive Self-Supervised Detection" algorithm developed by [13]. Using this algorithm, we locate boxes that bound character faces in each panel. We crop these bounding boxes to obtain the character faces. The extracted faces serve as input to our face clustering pipeline. The authors of the algorithm have $nms\_thold = 0.4$ and $conf\_thold = 0.65$ as default confidence thresholds. In addition to the default thresholds, we experiment with $nms\_thold = 0.45$ and $conf\_thold = 0.7$. We observe 967 fewer detections with the latter configuration, which amounts to a relative difference of -1.23%. In our clustering experiments, we use the faces detected with the latter configuration.

3.2 Face Clustering Pipeline

Feature Extraction

We investigate five different feature extraction techniques. These are shown in Table 1. Our motivation for selecting these features is to cover a wide range of approaches to feature extraction and to compare older and simpler methods with newer and more complex ones. HOG features are adept at detecting edges, whereas LBP and ORB features capture corners well. Color histograms contain the color distribution in an image, which is information that none of HOG, LBP, and ORB use. Finally, SimCLR uses a deep convolutional neural network to learn latent representations of images based on a contrastive loss [21]. We believe that these methods collectively cover a wide range of approaches to feature extraction. For the interested reader, we provide more information on the inner workings of these methods in section 2.

Features Extraction Method
Histogram of Oriented Gradients (HOG)
Local Binary Patterns (LBP)
Oriented FAST and Rotated BRIEF (ORB)
Color Histograms
SimCLR

Table 1: The feature extraction methods under investigation.

For the computation of HOG and LBP features, we use the implementations of the scikit-image library [24]. For ORB features, we use the OpenCV implementation [25] to calculate key points and descriptors and use the Fisher vector implementation of [26] to obtain the final feature vectors. We use the MMSelfSup library [27] for the training of SimCLR.

With the exception of SimCLR, we apply grid search to optimize the hyperparameters of the feature extraction methods. We share the hyperparameter configurations we explore in Appendix A. We skip hyperparameter optimization for SimCLR due to time constraints and use the configuration from the original paper. This configuration is placed in Appendix A as well. We also share the complete MMSelfSup configuration file we use in our GitHub repository.

Preprocesing

We resize every image before feature extraction to ensure that images yield equal-sized vectors for a given feature extraction technique. Different dimensions are needed for different techniques due to different input requirements. HOG features require the input size to be a multiple of the cell and block sizes. For SimCLR, the dimensions of the input images must match the input layer of the ResNet50 architecture. LBP and Color Histograms do not have a strict requirement on the input dimensions. Their output dimensions, however, scale linearly with the input dimensions. Therefore, we use small dimensions for both. For ORB features, larger dimensions are needed as the computation itself applies downscaling to the input images. The exact dimensions used for each feature are shown in Table 2.

We also convert the images to grayscale for the HOG, LBP, and ORB features. For SimCLR, we additionally apply color normalization to each channel based on $\frac{x_c - \mu_c}{\sigma_c}$ where x is a pixel value in channel $c \in red, green, blue$. We experiment with the color statistics of our training dataset and those of the ImageNet dataset. Our experiments show that ImageNet statistics work better.

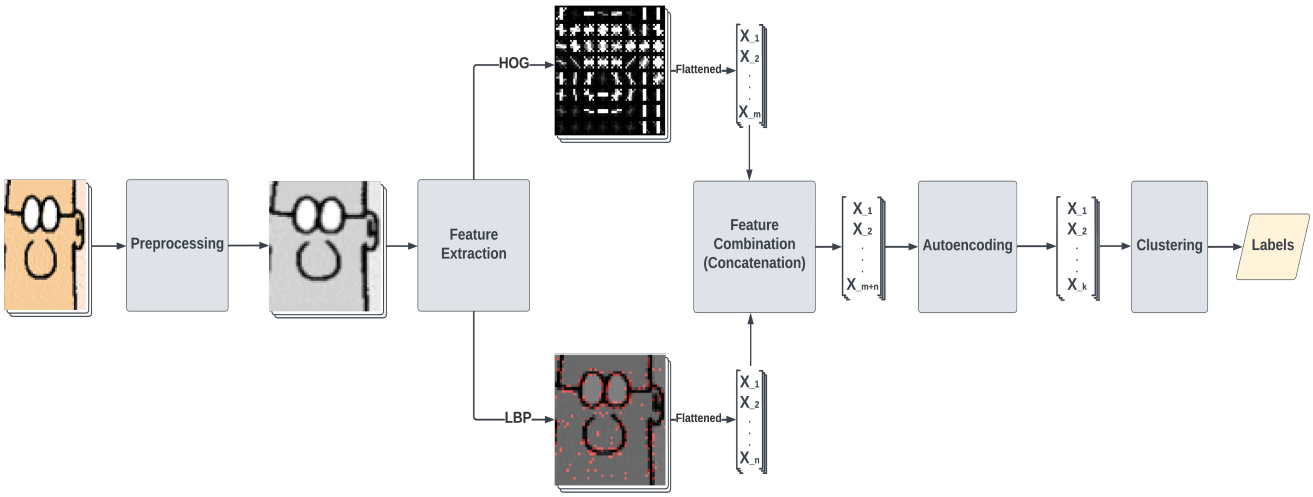

Figure 1: The face clustering pipeline with HOG and LBP features as example extraction techniques. These techniques can be changed with other ones.

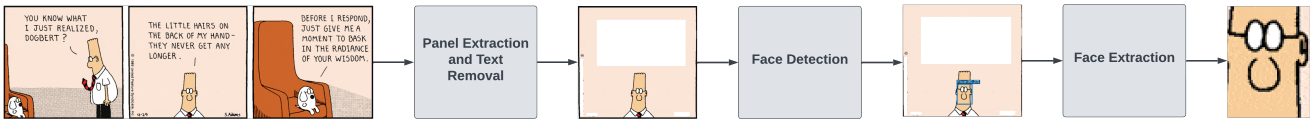

Figure 2: The face extraction pipeline for extracting faces of characters from comics.

Feature	Dimensions
HOG	(64, 64)
LBP	(48, 48)
ORB	(256, 256)
Color Hist.	(48, 48)
SimCLR	(224, 224)

Table 2: The dimensions used for resizing images during preprocessing per feature.

Feature Combination and Autoencoding

As mentioned previously, each feature extraction technique we investigate focuses on a different aspect of an image. Therefore, we hypothesize that we can improve the discriminatory power of feature extraction methods by combining their feature vectors for a given image. A combination of feature vectors may hold more information than its parts.

When we use SimCLR for feature extraction, we rely on a deep convolutional neural network to learn representations. The underlying CNN can learn to detect multiple shapes (edges, corners, etc.) at the same time. Furthermore, the input to this CNN is RGB images, which also allows it to learn from color information. Thus, we think that SimCLR would not benefit from the complementary nature of feature combination, so we exclude it from the combination experiments.

For each feature extraction technique, we use the optimal hyperparameter configuration for the combination experiments. We define the optimal configuration as the one that yields the lowest Davies-Bouldin score using K-Means++ clustering. We use Davies-Bouldin for this purpose for two reasons. First, it does not require external information. As a result, we maintain the unsupervised nature of our face clustering approach and avoid information leakage from test data. Second, it incorporates two aspects of clustering we are looking to maximize: separation between clusters and cohesion among clusters.

After identifying the optimal configurations, we combine the feature vectors of the methods by concatenating the feature vectors that belong to the same image. If a method fails to extract features from an image, the image is excluded from the combined features. We have 4 feature extraction methods to combine: HOG, LBP, ORB, and Color Histograms. We experiment with every 2-combination these 4 features have, giving us 6 combined features in total: HOG-LBP, HOG-ORB, HOG-Color Histogram, LBP-ORB, LBP-Color Histogram, and ORB-Color Histogram.

Finally, we experiment with autoencoding the combined feature vectors. We hypothesize that autoencoding can capture the correlations between vectors of different feature methods and also reduce the dimensionality of the vectors, which can boost the clustering performance. We use two different autoencoding methods: randomized PCA and neural network autoencoders, also shown in Table 3. We experiment with different configurations and latent representation sizes for both methods. The explored configurations can be found in Appendix B.

Autoencoding Method
Randomized Principal Component Analysis (PCA)
Neural Network Autoencoder

Table 3: The autoencoding methods under investigation.

For randomized PCA, we use the implementation of the scikit library [28]. For the implementation of neural autoencoders, we use tensorflow [29].

Clustering
We experiment with two clustering algorithms: K-Means++ [5] and Approximate Rank-Order Clustering (AROC) [2], also shown in Table 4.

Clustering Method
K-Means++
Approximate Rank-Order Clustering (AROC)

Table 4: The clustering methods under investigation.

We use K-Means++ for two purposes: hyperparameter tuning of feature extraction methods and as a baseline for a comparison with AROC. We justify our choice of K-Means++ for hyperparameter tuning by noting that its assumptions and properties match our data and purposes. First, we experiment with a multitude of configurations for every feature extraction method. Therefore, we benefit from the scalability and speed of K-Means++ when run with mini-batches. Second, K-Means++ assumes that the clusters are convex. We know that the faces of the comic characters are consistently similar. If we take the average face of a character as a point in the feature space, the various faces of the same character form a hypersphere around this point. Thus, the points belonging to the same character have a convex shape. For these reasons, we believe that K-Means++ fits our use case.

K-Means++ requires the number of clusters prior to training. We set the number of clusters to the number of main characters we identify in our experimental dataset of Dilbert comics. This decision is based on the following observation about a random sample we took from our dataset. Out of a random sample of 1500 face images, only 264 did not belong to the main characters. Furthermore, we were unable to identify these characters using character lists from Wikipedia[4] and Dilbert Wiki[5]. Thus, we conclude that these characters are minor and, unlike the main characters, do not persist throughout 35 years of publication.

AROC is an interesting choice for three reasons. First, it is a hierarchical clustering technique that can identify noisy samples. We achieve this by adding a new parameter to the algorithm called "min_samples." Using this parameter, we label the members of a cluster as noisy if the number of samples in that cluster is less than "min_samples." Second, AROC has been shown to perform well in the task of clustering real human faces [2]. Finally, since it is based on approximate nearest neighbors, it is computationally efficient and can scale to a large number of faces.

[4]https://en.wikipedia.org/wiki/Dilbert#Characters

[5]https://dilbert.fandom.com/wiki/Category:Characters

Unlike K-Means++, however, AROC has two hyperparameters that must be empirically tuned before use. These are called "n_neighbors" and "threshold." The former determines the number of approximate nearest neighbors to keep per data point, and the latter indicates the distance threshold under which two clusters are merged. The grid search strategy we use for these hyperparameters can be found in Appendix C. To limit an exponential increase in the number of configurations we experiment, we use AROC with a select set of feature extraction methods. These are the best-performing single feature and the best-performing combined feature. Our aim is to see if we can improve on the K-Means++ baseline with AROC using our custom parameter "min_samples."

For K-Means++, we use the implementation of the scikit library [28]. For AROC, we rely on the implementation[6] by the authors of [2]. We further modify the implementation to add our custom "min_samples" parameter.

4 Experimental Setup and Results

We dedicate this section to our experiments. First, we describe our evaluation approach including the data and the metrics used. We follow that with the results of our experiments.

4.1 Data

We use Dilbert Comics to evaluate the performance of our face clustering pipeline. The comics span the years 1989 to 2023, and there are 12,384 comic strips in total. Using the face extraction pipeline described in section 3, we extract 77,768 face images. These images form our experimental dataset. Figure 1 contains an example face image as input of the clustering pipeline.

We identify 12 main characters in Dilbert Comics and base our external evaluation on them. To obtain our ground-truth labels, we sample 1,500 random images from our dataset and manually label them. After removing images that do not correspond to any of the main characters, we are left with 1,236 labeled face images. The names and count distribution of these characters are given in Table 5.

4.2 Evaluation Metrics

Our evaluation of clustering performance is twofold: internal and external evaluation. We use the silhouette score [30] and the Davies-Bouldin [31] score as our internal metrics. For external evaluation, we use pairwise precision, recall, and F1 score. The ground-truth labels used in the calculation of external metrics are shown in Table 5.

4.3 Results

The exact hyperparameter configurations of the reported results and the grid search strategy we employ for all feature extraction, autoencoding, and clustering methods displayed in this subsection can be found in Appendix A, B, and C, respectively.

[6]https://github.com/KunpengWang/approximate-rank-order-clustering

Character	Dilbert	Boss	Dogbert	Wally	Alice	Carol	Asok	Ratbert	Catbert	Tina	Garbageman	Bob	Total
Count	409	235	175	142	110	39	37	36	21	17	11	4	1,236

Table 5: The distribution of characters in the labelled face images.

Comparison of Feature Extraction Methods

In Table 6, we report the performance of each feature extraction method based on K-Means++ clustering. For each feature, we report the hyperparameter configuration that maximizes the F1 score. As a baseline, we use a random feature that assigns values sampled from the standard normal distribution to each dimension.

We observe that Color Histograms outperform all other features by a clear margin with an F1 score of 0.6. Then comes HOG, LBP, and SimCLR, all of whom attain similar F1 scores. The worst performing feature is ORB. The results indicate that color information is crucial for differentiating between comic character faces.

Interestingly, ORB features yield the best Davies-Bouldin score, but they have the lowest F1 score excluding the random feature baseline. This inverse relationship between Davies-Bouldin and F1 scores can also be observed in LBP, albeit in the opposite direction. Having the worst Silhouette and Davies-Bouldin scores, LBP is located in the middle of the pack in terms of the F1 score. Based on Table 6, Davies-Bouldin and F1 scores have a correlation coefficient of 0.023. It is clear that there is no significant correlation between Davies-Bouldin and F1 scores. This suggests that internal clustering metrics do not necessarily capture information retrieval capacity.

Effectiveness of Feature Combination

Next, we test the effectiveness of combining features. We have 4 features to combine, and we experiment with 2-combinations. This gives us 6 combined features in total: HOG-LBP, HOG-ORB, HOG-Color Histogram, LBP-ORB, LBP-Color Histogram, and ORB-Color Histogram. For each feature, we select the hyperparameter configuration that yields the lowest, i.e. the best, Davies-Bouldin score. Additionally, we experiment with autoencoding the combined feature vectors before clustering them. Hence, we have 3 variations of every combination: combination without reduction, combination and reduction with PCA, and combination and reduction with neural autoencoders. Figure 3 plots the F1 scores of these variations based on K-Means++ clustering. We also plot the F1 scores of the individual features that make up the combined features for comparison.

From Figure 3, we observe that combining features does not guarantee an improvement in performance. For example, combinations that contain LBP perform very similar to LBP itself. We attribute this to the fact that LBP features consist of 2304 dimensions, which is three times larger than the method with the second highest dimensionality, measuring 768 dimensions. Therefore, the LBP part of the combined features seems to dominate during clustering.

On the other hand, the feature that benefits the most from combination is Color Histograms. When we combine Color Histograms with HOG and ORB features, we observe an increase in performance, with and without autoencoding. This is an indication that combining color and shape information can improve the clustering of comic character faces.

With the exception of the LBP-ORB combination, we observe a general increase in accuracy after autoencoding the combinations. However, the increase is marginal in some cases. ORB-Color Histogram seems to benefit the most from autoencoding, with its neural network-reduced version achieving the highest F1 score of 0.609. This score is also higher than that of any single feature.

Comparison of Clustering Algorithms

The final component of our investigation concerns the choice of clustering technique. In addition to K-Means++, we experiment with an agglomerative hierarchical clustering algorithm called Approximate Rank-Order Clustering (AROC). We use the following features to experiment with AROC:

- The best performing single feature: Color Histograms
- The best performing combined feature: ORB-Color Histogram reduced with a neural network autoencoder

In Table 7, we compare the performance of AROC with that of K-Means++. The displayed results of AROC are obtained with "min_samples" parameter set to 100 and the rest of the hyperparameters tuned using a grid search. We also experiment with lower "min_samples," but the number of clusters grows exponentially. This renders manual labeling of clusters infeasible. During the evaluation, we remove the noisy images from the test images to keep the evaluation based only on the clustered data points.

Our first observation is that there is a positive correlation between the number of noisy samples removed and the precision of the clustering. In both features, AROC clustering identifies approximately 50,000 data points as noisy. In turn, the precision of both features increases dramatically, with an increase of 77.6% in Color Histograms and 26.3% in ORB-Color Histograms. This is expected, as the removal of noisy samples allows AROC to only cluster data points that are tightly grouped.

In contrast, we observe a decrease in recall by 13.4% in Color Histogram and 31.9% in ORB-Color Histograms when we swap K-Means++ with AROC. This is the result of an increase in the number of clusters. Recall is defined as the ratio of the number of same-class pairs that are placed in the same cluster to the total number of same-class pairs. Therefore, a higher number of clusters reduces recall by definition.

As F1 depends on both precision and recall, the results suggest that AROC does not guarantee an increase in F1. However, we obtain the highest F1 score of our experiments using AROC on Color Histograms, with an F1 score of 0.752 on 20,988 face images. A precision of 0.886 gives further credence to the feasibility of collectively labeling clusters.

Feature	Image Count	Feature Dimensions	Silhouette	Davies-Bouldin	Precision	Recall	F1
HOG	77,768	216	0.139	2.435	0.537	0.353	0.426
LBP	77,768	2304	0.002	6.532	0.345	0.515	0.413
ORB	67,482	640	0.248	**1.188**	0.230	0.200	0.214
Color Hist.	77,768	768	**0.314**	1.243	**0.499**	**0.754**	**0.600**
SimCLR	77,768	2048	0.097	2.600	0.374	0.426	0.399
Random	77,768	10	0.059	2.177	0.188	0.082	0.114

Table 6: Performances of the features based on K-Means++ clustering. For each feature, we report the configuration that maximizes F1.

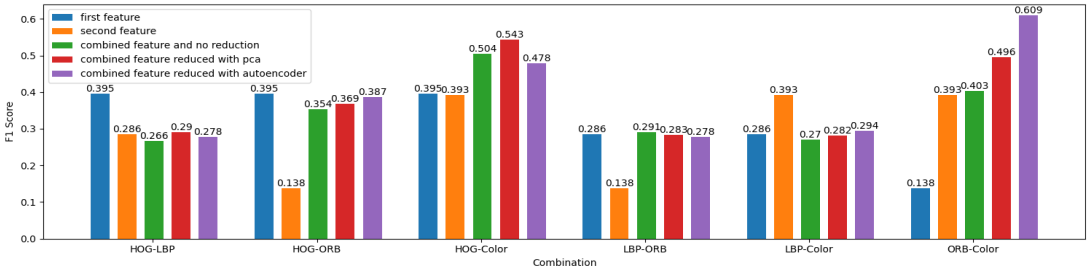

Figure 3: F1 Scores of the K-Means++ clustering of combined features, with and without reduction, compared to the scores of the individual features.

5 Discussion

In this section, we revisit the research questions posed in section 1 and use our experimental data to answer them.

How effective are different feature extraction methods for comic face clustering?

In our experiments, Color Histograms outperform other methods by a significant margin. Using K-Means++, Color Histograms achieve an F1 score of 0.6, which is 0.174 higher than that of the second best method. With the exception of SimCLR, every method in our experiments operates with grayscale images. This emphasizes the importance of incorporating color information when clustering faces of comic characters. Since comics usually use a wider range of colors in the illustration of characters compared to images of real humans, color becomes a powerful distinguishing characteristic for comic face clustering.

ORB features, on the other hand, perform the worst. We note that ORB has the most hyperparameters out of all the methods we test. Its inferior performance can be the result of limited hyperparameter optimization. Nevertheless, this is a disadvantage of ORB, as it is outperformed by simpler HOG and LBP methods. Another complex approach, SimCLR also performs worse than HOG and LBP. Therefore, we conclude that simpler feature extraction methods perform better for comic face clustering.

We also observe a lack of correlation between the internal and external clustering evaluation metrics. This suggests that internal metrics may not capture the information retrieval capacity of feature extraction methods. Therefore, consideration of external metrics is essential for an accurate assessment of feature extraction methods for comic face clustering.

Can we improve the discriminative power of feature extraction methods by combining their feature vectors and autoencoding them?

We observe that combining features does not consistently improve accuracy. Some combinations outperform their parts, whereas the opposite is the case for other combinations. However, our results demonstrate that the combination of color and shape information improves clustering accuracy. This is evidenced by the combinations of HOG-Color Histograms and ORB-Color Histograms outperforming their single features.

Autoencoding the combined features, excluding the LBP-ORB combination, generally leads to a slight performance increase. Although the improvement is modest in some cases, autoencoding shows potential in enhancing the discriminative power of combined features. The most effective combination involves ORB and Color Histograms with a neural network autoencoding. This combination outperforms all other combinations in clustering comic character faces.

Interestingly, a combination can be dominated by a feature if there is a large difference between the dimensionality of the combined features. This is illustrated by the combinations that involve LBP. The mean absolute difference between the F1 scores of LBP and its combinations is 0.0084. We connect this small difference to the fact that LBP features have 2304 dimensions, which is three times larger than the method with the second highest dimensionality.

Can we outperform K-Means++ using Approximate Rank-Order Clustering and noisy sample detection?

Our results indicate that Approximate Rank-Order Clustering (AROC) does not guarantee a superior performance compared to K-Means++. However, it does offer the advantage of increased precision when a suitable "min_samples" parameter is used to remove noisy samples. This leads to purer clusters.

Feature	Clustering	# Img.	# Clustered Img.	# Test Img.	# Clusters	Precision	Recall	F1
Color Hist.	K-Means++	77,768	77,768	1,236	12	0.499	**0.754**	0.600
Color Hist.	AROC	77,768	20,988	317	35	**0.886**	0.653	**0.752**
ORB-Color	K-Means++	76,885	76,885	1,236	12	0.578	**0.643**	**0.609**
ORB-Color	AROC	76,885	22,441	411	100	**0.730**	0.438	0.547

Table 7: A comparison of K-Means++ and AROC.

In our experiments, we achieve our highest pairwise precision of 0.886 when we cluster Color Histograms using AROC with 100 minimum samples in each cluster.

A notable distinction between AROC and K-Means++ is the resulting number of clusters. AROC generally results in a higher number of clusters compared to K-Means++. In turn, AROC obtains a lower recall metric. This suggests that AROC's tendency to generate more fine-grained clusters comes at the expense of comprehensive cluster coverage of comic faces.

Among all feature extraction, combination, and clustering methods we evaluate, the best overall results are achieved using AROC on Color Histograms with 100 minimum samples per cluster. This configuration yields an F1 score of 0.752 by clustering 20,988 face images out of 77,768 into 35 clusters.

6 Responsible Research

Potential Bias in Results

According to a report by The Alan Turing Institute, racial and gender bias is a common problem in facial recognition systems [32]. The authors argue that the racism and sexism that exist in the training datasets of these systems carry over as bias in the results. We believe that a similar problem can also occur in comic face clustering. Certain datasets might be skewed toward white male characters. Dilbert Comics is an example of this, where male characters are seen more often than female ones. Racial representation is worse, with the only person of color human character being Asok. Thus, the results we obtain in this paper using Dilbert Comics might carry a bias that neglects people of color. Future research should always aim to have a fair representation of people in the training datasets to ensure unbiased results.

Reproducibility of the Results

Reproducibility is fundamental for conducting research. It motivates researchers to assess the robustness of their results and allows fellow researchers to verify them. To enable the reproducibility of our work, we publish our source code on GitHub[7] including our experiment scripts. With the help of our documentation and usage guide, interested parties can reproduce our results. In addition, we structured our code base in a modular way so that other researchers can use parts of it in their research and extend it if necessary.

Copyright Considerations

Comics are the intellectual property of their creator. Therefore, their use is bound by copyright laws. In our research, we use a public Dilbert Comics dataset compiled from publicly available comics on dilbert.com[8]. Therefore, we do not risk any copyright violation. However, our work cannot and should not be used for commercial purposes.

7 Conclusions and Future Work

In this work, we examine the effectiveness of face clustering on comic characters. To conduct our examination, we implement two pipelines. The first automatically extracts character faces from comic strips. The second helps us assess the effectiveness of various feature extraction and clustering techniques. We also investigate the effectiveness of combining different feature extraction methods to improve clustering performance. Finally, we experiment with the removal of noisy samples to increase the accuracy of the clustering. Based on our experiments with Dilbert Comics, we show that

- Using color information is crucial for accurate comic face clustering.
- Combining feature extraction methods does not guarantee an increase in clustering accuracy. However, combining color with shape features does improve accuracy.
- Autoencoding the feature vectors generally enhances the clustering performance. However, the increase is marginal in most cases.
- Removal of noisy samples with hierarchical clustering can drastically increase the precision of clustering and result in purer clusters.

We combine these findings to achieve an F1 score of 0.752 on character faces extracted from Dilbert Comics. To achieve this result, we use Color Histograms extracted from 77,768 face images. Then, we use Approximate Rank-Order Clustering to remove noisy samples and cluster the remaining 20,988 face images into 35 clusters with a precision of 0.886.

In terms of future work, our methodology should be tested with other comic franchises. This can allow us to gauge the robustness of our results. Moreover, our face image dataset consists of images with fairly small dimensions. A dataset with larger images might result in higher clustering accuracy. Further improvements can be achieved by applying face alignment prior to feature extraction. Simultaneous training of feature extraction and clustering is also an interesting avenue for improvement.

[7] https://github.com/artunboz/bachelor-research

[8] Until 2022, all Dilbert comics were listed on this website. They were later removed, possibly due to the backlash the author received. For more information: https://edition.cnn.com/2023/02/25/business/dilbert-comic-strip-racist-tirade/index.html

References

[1] Y. Shi, C. Otto, and A. K. Jain, "Face clustering: Representation and pairwise constraints," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1626–1640, 2018.

[2] C. Otto, D. Wang, and A. K. Jain, "Clustering millions of faces by identity," 2016.

[3] B. Proven-Bessel, Z. Zhao, and L. Chen, "Comicgan: Text-to-comic generative adversarial network," 2021.

[4] K. Takayama, H. Johan, and T. Nishita, "Face detection and face recognition of cartoon characters using feature extraction," in *Proceedings of the IIEEJ Image Electronics and Visual Computing Workshop 2012*, November 2012.

[5] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (USA), p. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[6] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 343–347, 2014.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] H. Yanagisawa, T. Yamashita, and W. Hiroshi, "Manga character clustering with DBSCAN using fine-tuned CNN model," in *International Workshop on Advanced Image Technology (IWAIT) 2019* (Q. Kemao, K. Hayase, P. Y. Lau, W.-N. Lie, Y.-L. Lee, S. Srisuk, and L. Yu, eds.), vol. 11049, p. 110491M, International Society for Optics and Photonics, SPIE, 2019.

[9] Z. Zhang, Z. Wang, and W. Hu, "Unsupervised manga character re-identification via face-body and spatial-temporal associated clustering," 2022.

[10] N. Aldahoul, H. A. Karim, M. H. L. Abdullah, A. S. B. Wazir, M. F. A. Fauzi, M. J. T. Tan, S. Mansor, and H. S. Lyn, "An evaluation of traditional and cnn-based feature descriptors for cartoon pornography detection," *IEEE Access*, vol. 9, pp. 39910–39925, 2021.

[11] F. S. Khan, R. M. Anwer, J. van de Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez, "Color attributes for object detection," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3306–3313, 2012.

[12] N.-V. Nguyen, C. Rigaud, and J.-C. Burie, "Comic characters detection using deep learning," pp. 41–46, 11 2017.

[13] B. B. Topal, D. Yuret, and T. M. Sezgin, "Domain-adaptive self-supervised pre-training for face & body detection in drawings," 2023.

[14] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 582–585 vol.1, 1994.

[15] 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, pp. 886–893 vol. 1, 2005.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[17] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," vol. 3951, 07 2006.

[18] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 778–792, Springer Berlin Heidelberg, 2010.

[19] Sivic and Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, 2003.

[20] J. Sánchez, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, 12 2013.

[21] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[22] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, (San Francisco, CA, USA), p. 3–10, Morgan Kaufmann Publishers Inc., 1993.

[23] M. Styczen, "Automated text-image comic dataset construction," 2021.

[24] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, jun 2014.

[25] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[26] J. Rothfuss, "fishervector." https://github.com/jonasrothfuss/fishervector, 2018.

[27] M. Contributors, "MMSelfSup: Openmmlab self-supervised learning toolbox and benchmark." https://github.com/open-mmlab/mmselfsup, 2021.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[30] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[31] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[32] D. Leslie, "Understanding bias in facial recognition technologies," tech. rep., The Alan Turing Institute, 2020.

A Feature Extraction Configurations

This section contains the hyperparameter optimization search space and the optimal parameters found for each feature extraction method. We applied a grid search on the Cartesian product of the sets found in the first row of each table.

A.1 Histogram of Oriented Gradients (HOG)

	orientations	pixels per cell	cells per block	block normalization
Search Spaces	{6, 9}	{(8, 8), (16, 16)}	{(2, 2), (3, 3)}	{L1, L1-sqrt, L2, L2-Hys}
Optimal for F1	6	(16, 16)	(3, 3)	L2
Optimal for Davies-Bouldin	6	(16, 16)	(3, 3)	L2-Hys

Table 8: Hyperparameter search space and the optimal parameters for HOG feature.

A.2 Local Binary Pattern (LBP)

	P	R	variant
Search Spaces	set to $8 * R$	{1, 2, 3}	{basic, rotation-invariant, uniform}
Optimal for F1	8	1	uniform
Optimal for Davies-Bouldin	24	3	rotation-invariant

Table 9: Hyperparameter search space and the optimal parameters for LBP feature.

A.3 Oriented FAST and Rotated BRIEF (ORB)

There are additional parameters of OpenCV's implementation of ORB feature. These parameters are set to the following:

- edge threshold: set equal to the patch size as advised by OpenCV
- first level: 0
- WTA_K: 2
- score type: Harris score
- fast threshold: 20

	# fisher components	# keypoints	# levels	patch size
Search Spaces	{10, 20}	{10, 20}	{6, 8}	{16, 24, 31}
Optimal for F1	10	10	8	31
Optimal for Davies-Bouldin	10	10	6	24

Table 10: Hyperparameter search space and the optimal parameters for ORB feature.

A.4 Color Histogram

	histogram size per channel
Search Spaces	{32, 64, 128, 256}
Optimal for F1	256
Optimal for Davies-Bouldin	128

Table 11: Hyperparameter search space and the optimal parameters for Color Histogram feature.

A.5 SimCLR

The configuration we use for SimCLR training is as follows. For the exact configuration file, refer to our code on GitHub.

- **Preprocessing** Color normalization with RGB means and standard deviations of (198.878, 167.418, 132.772) and (21.34, 25.105, 26.093)
- **Architecture**

  - **Backbone** ResNet50 with batch normalization
  - **Neck** MLP with a single hidden layer and ReLU activation
  - **Loss** Normalized Temperature-scaled Cross Entropy
- **Optimizer** Layer-wise Adaptive Rate Scaling (LARS)
  - **Learning Rate** 0.3
  - **Weight Decay** 1e-06
  - **Momentum** 0.9
- **Training Schedule**
  - **Batch Size** 32
  - **Epochs** 200
    * First 10 epochs: linear learning rate change with a factor of 0.0001
    * Last 190 epochs: cosine annealing learning rate change with maximum temperature of 190

- Normalized Temperature-scaled Cross Entropy - optimizer: type='LARS', lr=0.3, weight_decay=1e-06, momentum=0.9 - epochs: linear learning rate change with factor of 0.0001 in the first 10 epochs, then 190 epochs of cosine annealing learning rate with maximum temperature of 190 - default data augmentation, refer to paper - batch size 32

B Autoencoding Configurations

This section contains the autoencoding configurations we test in our experiments. We apply each configuration to every feature combination and report the ones that result in the highest F1 score.

B.1 Neural Network

Our neural network-based autoencoder architecture consists of symmetric and fully connected encoder and decoder. We experiment with using one and two hidden layers and a different number of neurons in each. The number of neurons in the last hidden layer represents the dimensions of the latent representation. We use ReLU activation for all layers except the output layer of the decoder, for which we use sigmoid. Additionally, we min-max normalize the input to the encoder as preprocessing. Finally, we use the Adam optimizer with Mean Squared Error loss and train for 30 epochs with a batch size of 256. Table 12 contains the number of layers and number of neurons we search, and Table 13 contains the optimal configurations for the F1 score per feature combination.

Config.	hidden layer 1	hidden layer 2
1	10	X
2	50	X
3	100	X
4	200	X
5	200	10
6	200	50
7	200	100

Table 12: The configurations tested for layer size and neuron count for the neural network autoencoder.

Feature	HOG-LBP	HOG-ORB	HOG-Color Hist.	LBP-ORB	LBP-Color Hist.	ORB-Color Hist.
Optimal for F1	Conf. 4	Conf. 1	Conf. 4	Conf. 3	Conf. 3	Conf. 4

Table 13: The optimal neural net autoencoder configuration based on F1 score. The configurations refer to Table 12

B.2 Principal Component Analysis

We use randomized Principal Component Analysis with 4 different number of components. These are [10, 50, 100, 200]. Table 14 contains the optimal number of components for the F1 score per feature combination.

C Clustering Configurations

This section contains the configurations of the clustering methods we use in our experiments.

Feature	HOG-LBP	HOG-ORB	HOG-Color Hist.	LBP-ORB	LBP-Color Hist.	ORB-Color Hist.
Optimal # Components	10	50	200	10	50	200

Table 14: The optimal number of components when applying PCA to reduce the dimensionality of feature combinations.

C.1 K-Means++

In all experiments involving K-Means++, the number of clusters is set to 12, which is the number of main characters we identify in Dilbert characters. The rest of the sklearn implementation parameters are set to the following:

- init: k-means++
- n_init: auto
- max_iter: 300
- tol: 1e-4
- random_state: None
- algorithm: lloyd

C.2 Approximate Rank-Order Clustering (AROC)

When using AROC, we apply two grid searches to optimize its hyperparameters. The first search is coarser and explores a wider range of values. The second search is more fine-grained, exploring the area around the best configuration found in the first search. The "min_samples" parameter is set to 100. We also experiment with values lower than 100, but the number of clusters becomes unmanageable. As an example, AROC clustering on Color Histograms with 1 min_samples result in 12536 clusters, and 10 min_samples result in 148 clusters when averaged over all configurations.

First Grid Search

The Cartesian product of the following values is tested:

- n_neighbours: [100, 200, 300, 400, 500]
- threshold: [0.1, 0.5, 1.0, 2.0, 5.0]

Second Grid Search

Color Histograms:

- n_neighbours: [190, 200, 210]
- threshold: [0.08, 0.1, 0.12]

ORB-Color Histograms:

- n_neighbours: [90, 100, 110]
- threshold: [0.4, 0.5, 0.6]

Optimal Hyperparameters

Color Histograms:

- n_neighbours: 190
- threshold: 0.12

ORB-Color Histograms:

- n_neighbours: 100
- threshold: 0.6