DS 203 Assignment E7

22b0321 : Aman Kumar Singh

22b0326 : Pratyush Ranjan

22b0432 : Sarthak Mishra

22b2525 : Samyak Pahade

SLIDESMANIA.COM

Streamlining Architectural Design with Data Science

- We are given 1183 bitmap images of building layouts, each 640 x 480 pixels.
- We have to analyze these images to identify design families, classify the complexity of layouts, and develop a system to retrieve designs based on specific parameters. We also have to explore ways to use this data to enhance design processes and productivity which will save a lot of time and effort.
- **Methods:** Image Feature Detection algorithms such as **Convolutional Neural Network (CNN)**, Clustering algorithms such as **K-Means**

SLIDESMANIA.COM

Table of contents

SLIDESMANIA.COM

Objectives

Design Family Clustering

Complexity Classification

Categorize building layouts into shape-based families

Analyze layouts to establish complexity criteria as Low Complexity, Medium Complexity, High Complex
 and automate their classification through machine learning.

SLIDESMANIA.COM

Objectives

Design Retrieval

Exploring Data Utilization

Develop a system to match architects' parameters such as dimensions, the layout area, and the permissible layout complexity
with relevant designs for efficient retrieval and future use

Explore other uses of image data, like trend prediction and design optimization

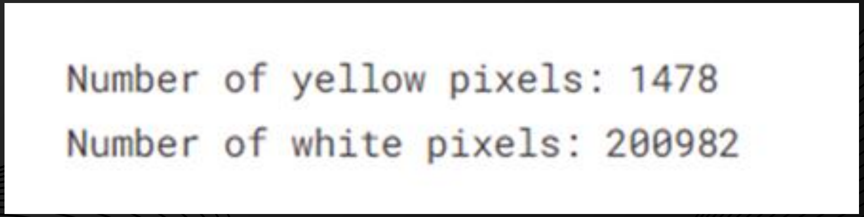
SLIDESMANIA.COM

Steps

1 Preprocessing

Basic Libraries:
- numpy
- cv2
- matplotlib

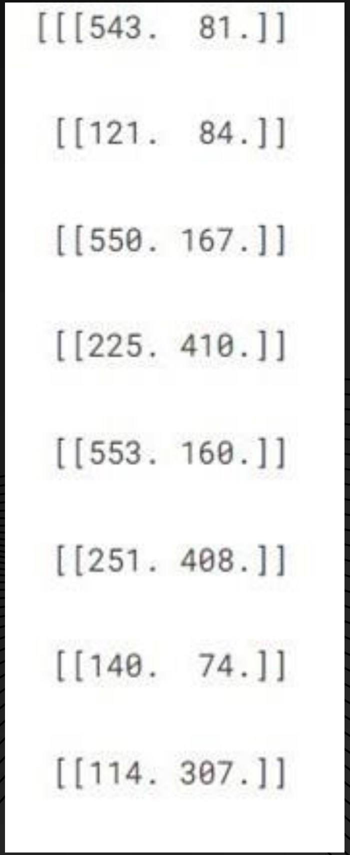
- At first, we preprocessed images to detect yellow and white pixels.
- On checking pixel count for 1181.jpg we get:


```
Number of yellow pixels: 1478
Number of white pixels: 200982
```

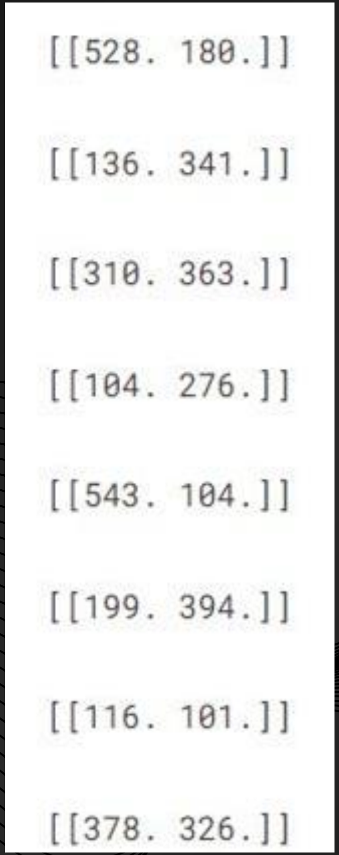
SLIDESMANIA.COM

1 Preprocessing

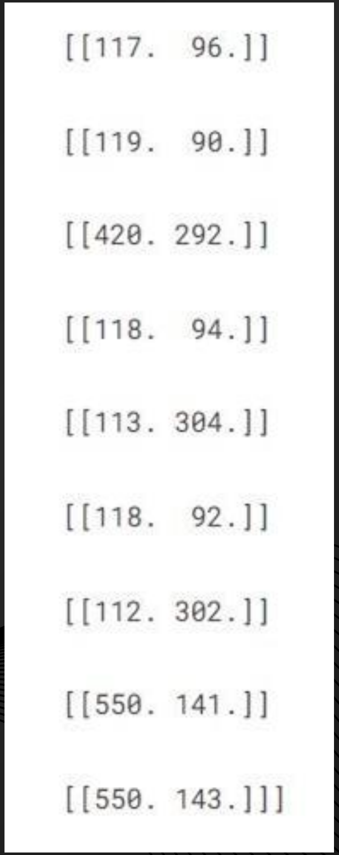
Steps



```
[[[543.   81.]]

 [[121.   84.]]

 [[550.  167.]]

 [[225.  410.]]

 [[553.  160.]]

 [[251.  408.]]

 [[140.   74.]]

 [[114.  307.]]
```



```
 [[528.  180.]]

 [[136.  341.]]

 [[310.  363.]]

 [[104.  276.]]

 [[543.  104.]]

 [[199.  394.]]

 [[116.  101.]]

 [[378.  326.]]
```



```
 [[117.   96.]]

 [[119.   90.]]

 [[420.  292.]]

 [[118.   94.]]

 [[113.  304.]]

 [[118.   92.]]

 [[112.  302.]]

 [[550.  141.]]

 [[550.  143.]]]
```

- Then, we preprocessed some images to determine the number of edges and corners.
- After implementing Canny corner detection code to extract corner coordinates from the image, our output was:

SLIDESMANIA.COM

1 Preprocessing

Inferences

The count of these points perfectly matched when we manually counted the edges. We performed similar counts for various other images as well.

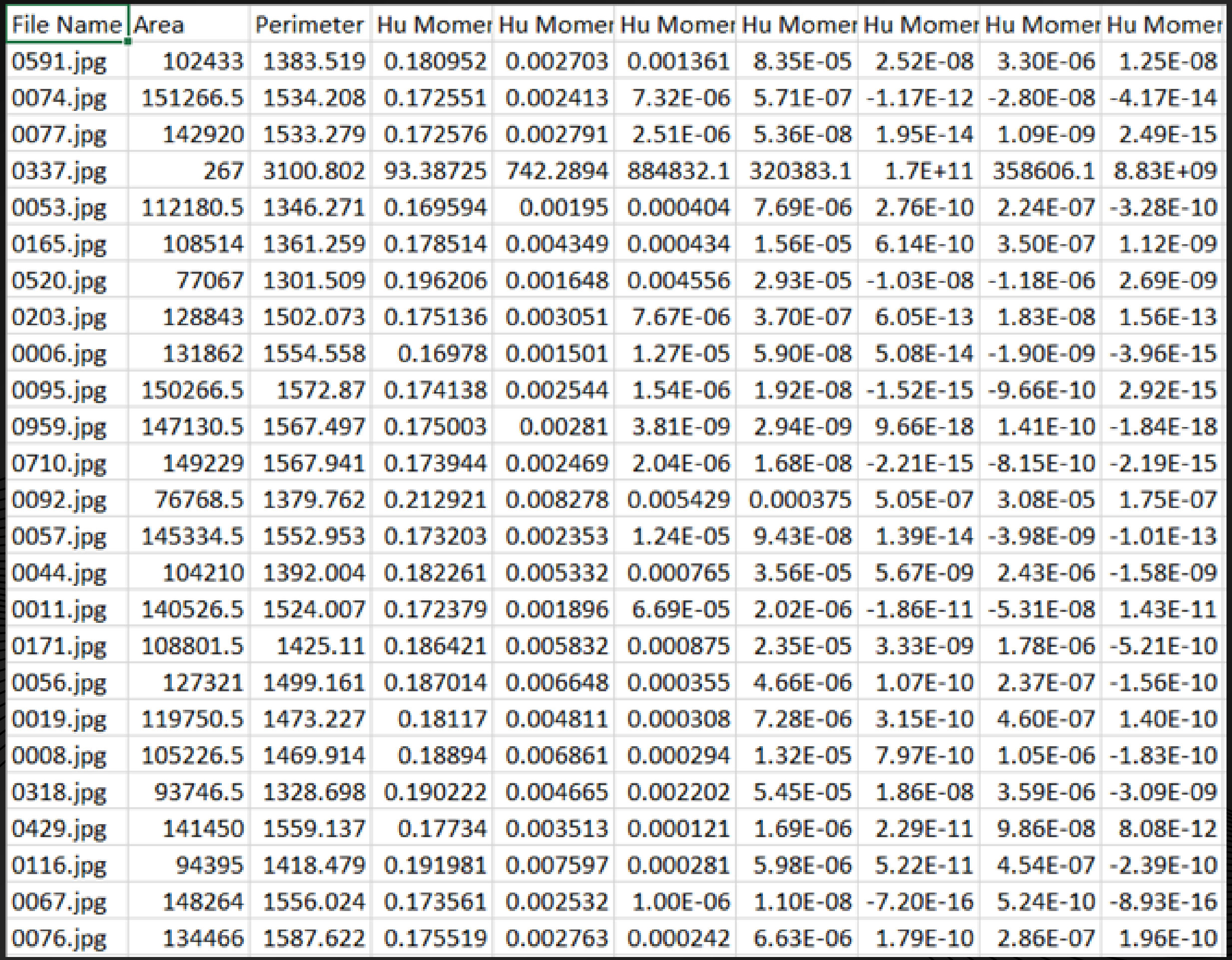
Problems

We tried two different ways, Gabor filtering and Canny edge detection, to count the edges, but no matter how we adjusted the settings, we couldn't get accurate results.

SLIDESMANIA.COM

Steps

2 Features Extraction



File Name	Area	Perimeter	Hu Momer	Hu Momer	Hu Momer	Hu Momer	Hu Momer	Hu Momer	Hu Momer
0591.jpg	102433	1383.519	0.180952	0.002703	0.001361	8.35E-05	2.52E-08	3.30E-06	1.25E-08
0074.jpg	151266.5	1534.208	0.172551	0.002413	7.32E-06	5.71E-07	-1.17E-12	-2.80E-08	-4.17E-14
0077.jpg	142920	1533.279	0.172576	0.002791	2.51E-06	5.36E-08	1.95E-14	1.09E-09	2.49E-15
0337.jpg	267	3100.802	93.38725	742.2894	884832.1	320383.1	1.7E+11	358606.1	8.83E+09
0053.jpg	112180.5	1346.271	0.169594	0.00195	0.000404	7.69E-06	2.76E-10	2.24E-07	-3.28E-10
0165.jpg	108514	1361.259	0.178514	0.004349	0.000434	1.56E-05	6.14E-10	3.50E-07	1.12E-09
0520.jpg	77067	1301.509	0.196206	0.001648	0.004556	2.93E-05	-1.03E-08	-1.18E-06	2.69E-09
0203.jpg	128843	1502.073	0.175136	0.003051	7.67E-06	3.70E-07	6.05E-13	1.83E-08	1.56E-13
0006.jpg	131862	1554.558	0.16978	0.001501	1.27E-05	5.90E-08	5.08E-14	-1.90E-09	-3.96E-15
0095.jpg	150266.5	1572.87	0.174138	0.002544	1.54E-06	1.92E-08	-1.52E-15	-9.66E-10	2.92E-15
0959.jpg	147130.5	1567.497	0.175003	0.00281	3.81E-09	2.94E-09	9.66E-18	1.41E-10	-1.84E-18
0710.jpg	149229	1567.941	0.173944	0.002469	2.04E-06	1.68E-08	-2.21E-15	-8.15E-10	-2.19E-15
0092.jpg	76768.5	1379.762	0.212921	0.008278	0.005429	0.000375	5.05E-07	3.08E-05	1.75E-07
0057.jpg	145334.5	1552.953	0.173203	0.002353	1.24E-05	9.43E-08	1.39E-14	-3.98E-09	-1.01E-13
0044.jpg	104210	1392.004	0.182261	0.005332	0.000765	3.56E-05	5.67E-09	2.43E-06	-1.58E-09
0011.jpg	140526.5	1524.007	0.172379	0.001896	6.69E-05	2.02E-06	-1.86E-11	-5.31E-08	1.43E-11
0171.jpg	108801.5	1425.11	0.186421	0.005832	0.000875	2.35E-05	3.33E-09	1.78E-06	-5.21E-10
0056.jpg	127321	1499.161	0.187014	0.006648	0.000355	4.66E-06	1.07E-10	2.37E-07	-1.56E-10
0019.jpg	119750.5	1473.227	0.18117	0.004811	0.000308	7.28E-06	3.15E-10	4.60E-07	1.40E-10
0008.jpg	105226.5	1469.914	0.18894	0.006861	0.000294	1.32E-05	7.97E-10	1.05E-06	-1.83E-10
0318.jpg	93746.5	1328.698	0.190222	0.004665	0.002202	5.45E-05	1.86E-08	3.59E-06	-3.09E-09
0429.jpg	141450	1559.137	0.17734	0.003513	0.000121	1.69E-06	2.29E-11	9.86E-08	8.08E-12
0116.jpg	94395	1418.479	0.191981	0.007597	0.000281	5.98E-06	5.22E-11	4.54E-07	-2.39E-10
0067.jpg	148264	1556.024	0.173561	0.002532	1.00E-06	1.10E-08	-7.20E-16	5.24E-10	-8.93E-16
0076.jpg	134466	1587.622	0.175519	0.002763	0.000242	6.63E-06	1.79E-10	2.86E-07	1.96E-10

We extracted features such as 7 Hu Moments, Area, and Perimeter of the image. These features serve as the basis for performing K-Means Clustering

SLIDESMANIA.COM

Hu Moments

- Hu Moments consist of seven scalar values derived from the central moments of an image.
- Hu Moments are used in image processing and computer vision for shape analysis and recognition.
- Central moments describe the distribution of intensity values within an image.
- Hu Moments provide scale, rotation, and translation invariant features for characterizing object shapes.
- They ensure robustness across different image conditions.
- Hu Moments are commonly used in pattern recognition tasks such as object recognition, shape matching, and image retrieval.

SLIDESMANIA.COM

Steps

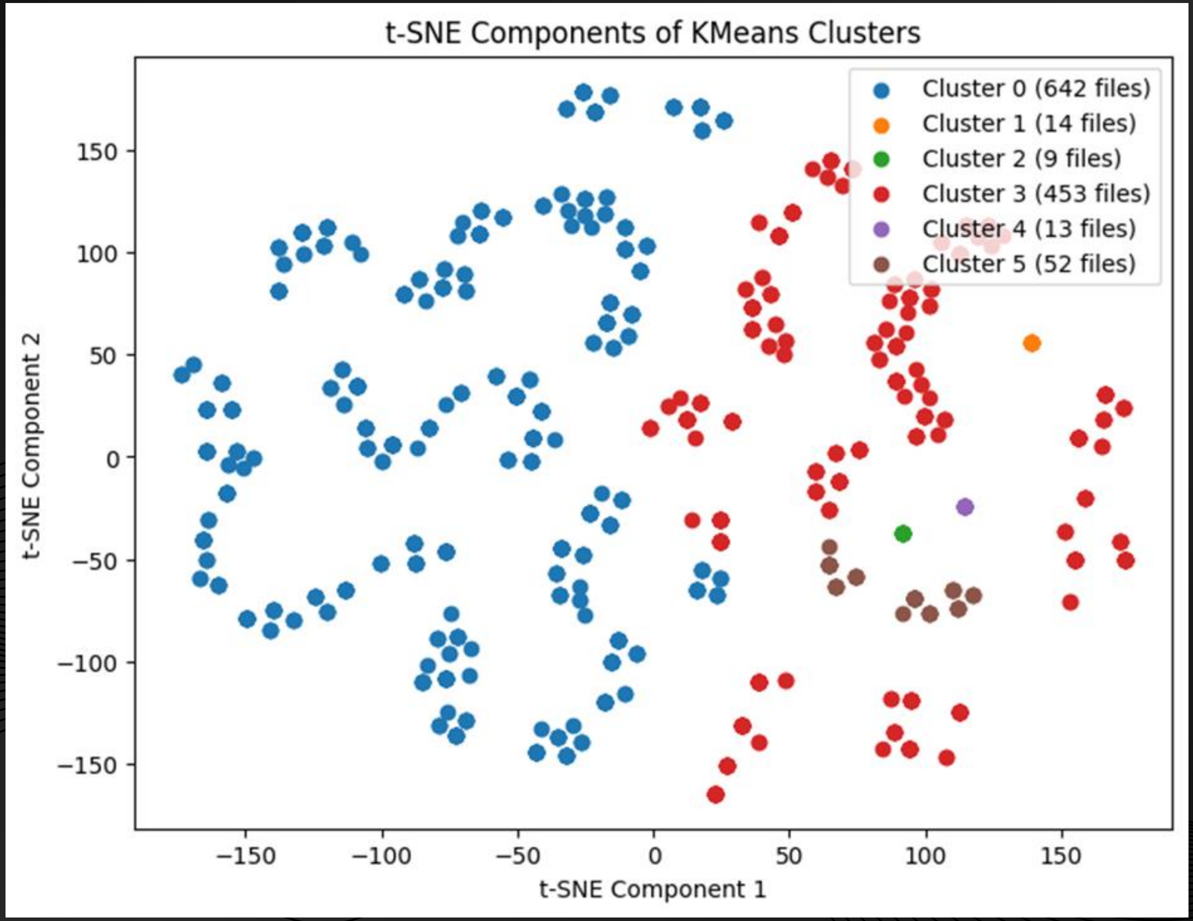
3 k-means clustering

- We initiated the k-means clustering algorithm to identify the maximum number of distinct clusters.
- While executing, a warning from the sklearn library alerted us about the limit on the maximum number of clusters that k-means can generate.
- In the first iteration, we observed a maximum of six clusters.

SLIDESMANIA.COM

Steps

4 t-SNE visualization



We then employed t-SNE visualization to assess the clustering of images and the distribution of images within each cluster.

Upon observing the graph, we notice that not all points are visible; it appears that some points are overlapping.

SLIDESMANIA.COM

Steps

5 Identifying distinct Clusters

- We re-ran k-means clustering on cluster 0 to identify the maximum number of distinct clusters.
- This time, we observed 100 distinct clusters, which appeared to be abnormal clustering behavior.
- To address this issue, we opted to categorize the images based on their clustering labels.
- We created subfolders for each cluster label and sorted images accordingly.
- Upon manual inspection of the 100 clusters within cluster 0, we found that each cluster comprised duplicate images.
- Each cluster contained approximately 7-8 identical images.

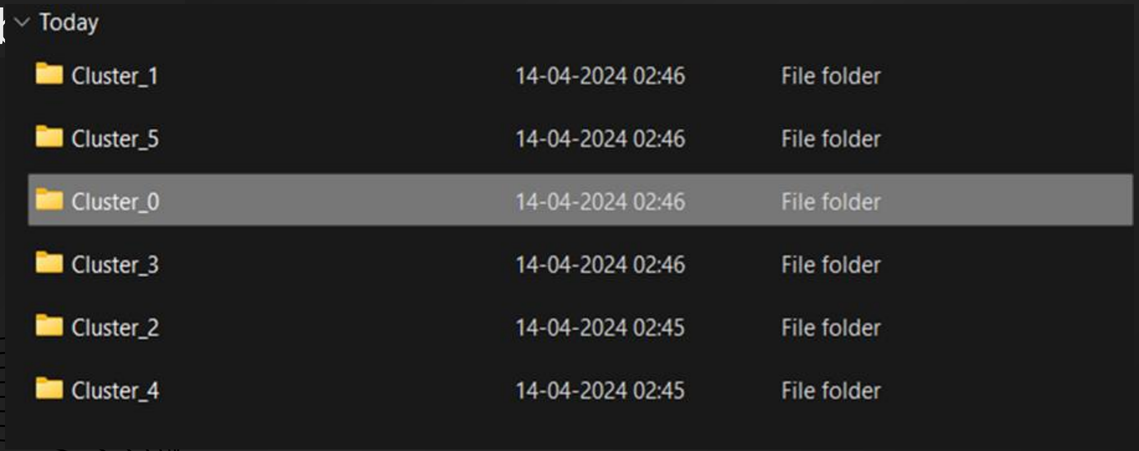
SLIDESMANIA.COM

Steps

5 Identifying distinct Clusters

On opening cluster_0 we get approx. 100 clusters

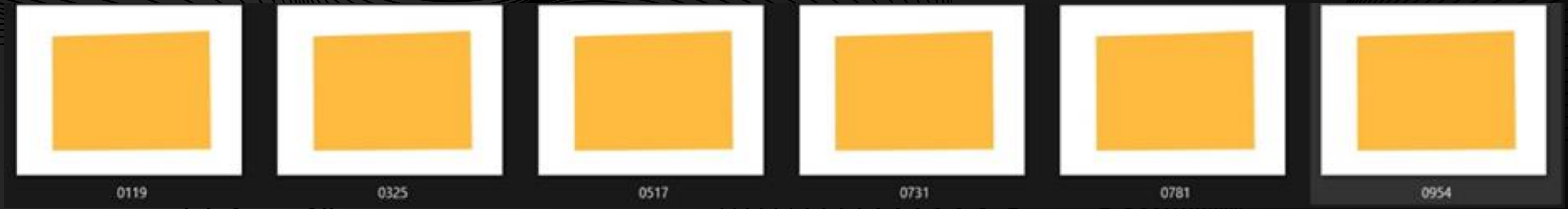
sul



Today

Cluster_1	14-04-2024 02:46	File folder
Cluster_5	14-04-2024 02:46	File folder
Cluster_0	14-04-2024 02:46	File folder
Cluster_3	14-04-2024 02:46	File folder
Cluster_2	14-04-2024 02:45	File folder
Cluster_4	14-04-2024 02:45	File folder

Then opening the subfolder for Cluster_20, the images appear as follows:



- We can see that the images are exactly identical.

SLIDESMANIA.COM

Steps

6 Removing duplicate images

- Initially, we decided to remove all duplicate images before re-running the k-means clustering algorithm.
- This process involves file hashing using SHA-256, where each image file in a directory receives a unique cryptographic hash.
- The hash is computed based on the content of the file, ensuring that files with identical content produce identical hash values.

SLIDESMANIA.COM

SHA – 256 algorithm

- The SHA-256 algorithm, part of the SHA-2 family, generates a 256-bit hash, providing a high level of uniqueness.
- By utilizing SHA-256 hashing, duplicate images can be efficiently identified solely based on their content, rather than relying on potentially unreliable factors like filenames or metadata.
- This method is robust and widely used for identifying duplicates across various types of files.

SLIDESMANIA.COM

6 Removing duplicate images

Dictionary Created

We've created a dictionary associating each image name with its corresponding hash value, computed using the SHA-256 algorithm. Subsequently, we eliminated all duplicates sharing the same hash value.

Inferences

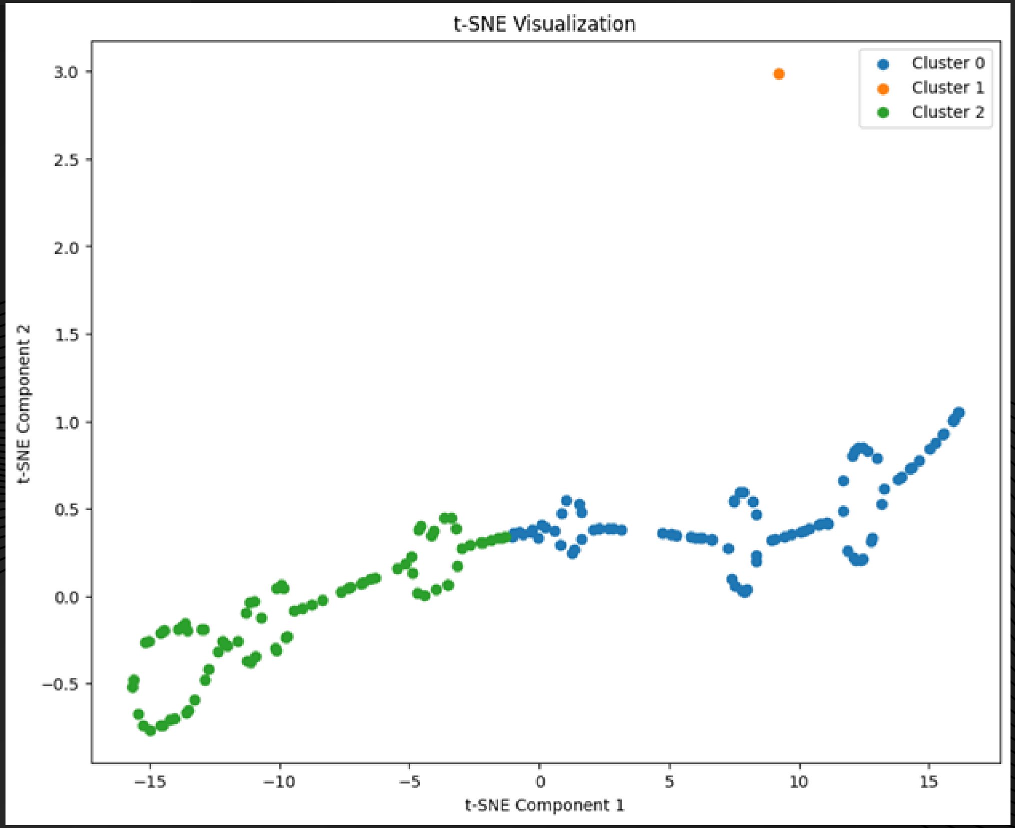
- Following the removal of duplicate images, our count decreased significantly from 1183 to just 173 images.
- Given the low number of images remaining, we opted not to downscale them to lower resolutions.

SLIDESMANIA.COM

Steps

7 Complexity Classification

To categorize images based on complexity (High, Low, and Medium), we set the desired clusters to 3. Below is the t-SNE plot depicting the results.



During the k-means clustering process on these images, we discovered that the maximum limit of clusters was 173, determined by features such as Hu moments, Area, and Perimeter of the image.

Silhouette Score: 0.6062498542605892
Davies-Bouldin Score: 0.3150324624909610

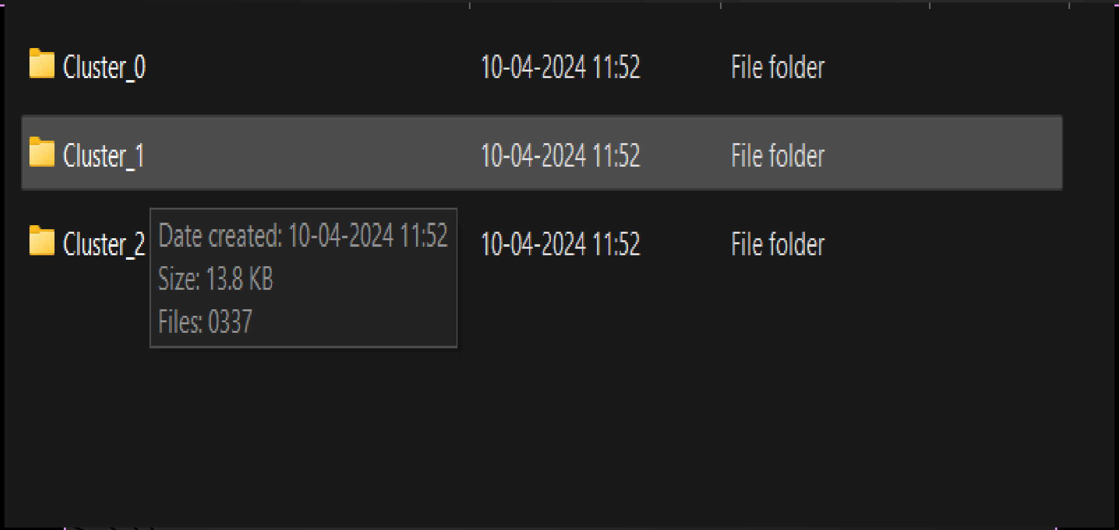
Cluster 0: 98
Cluster 1: 1
Cluster 2: 74

SLIDESMANIA.COM

Though the scores like Davis Bouldin and Silhouette Score were pretty good for above type of cluster, we again went on manually checking what images goes into these clusters. On checking we found that only one image was present in one of the cluster.

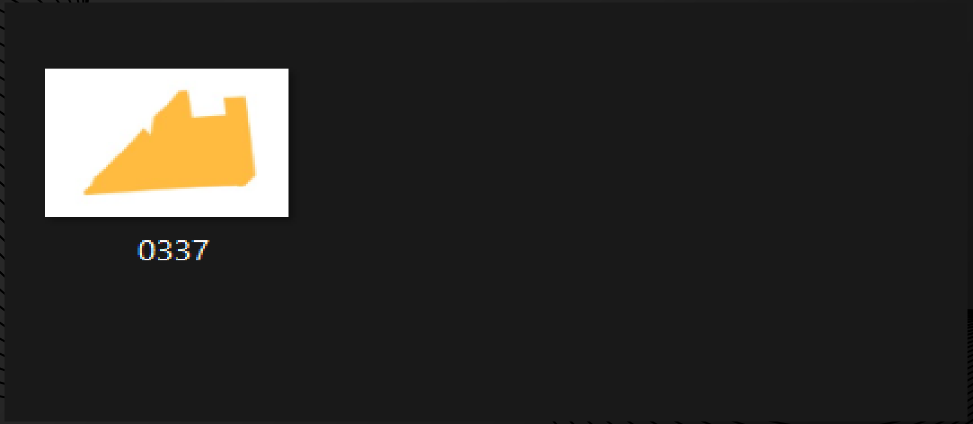
Silhouette Score

- Silhouette Score: This metric ranges from -1 to 1.
- A score closer to 1 suggests that the samples are distant from neighboring clusters.
- Conversely, a score closer to -1 indicates proximity to neighboring clusters.
- A score around 0 suggests overlapping clusters.



Cluster_0	10-04-2024 11:52	File folder
Cluster_1	10-04-2024 11:52	File folder
Cluster_2	10-04-2024 11:52	File folder

Date created: 10-04-2024 11:52
Size: 13.8 KB
Files: 0337



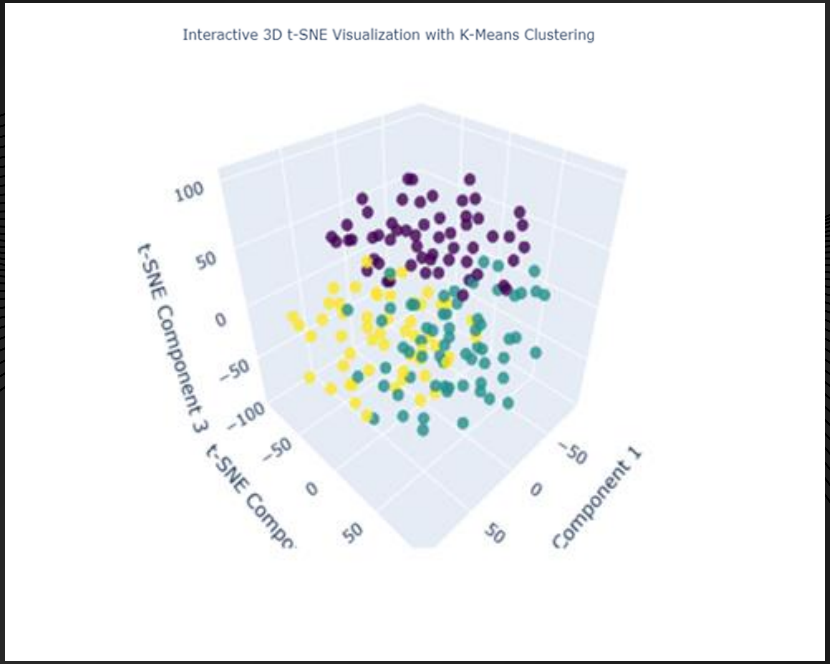
Davies-Bouldin Score

- The Davies-Bouldin score evaluates the average similarity between clusters.
- It measures the average similarity of each cluster with its most similar cluster.
- The score ranges from 0 to positive infinity.
- A lower Davies-Bouldin score indicates better clustering performance.
- A score of 0 represents the best possible outcome.

Steps

8 Manual Checking for Classification

Though the scores like Davis Bouldin and Silhouette Score were pretty good for this type of cluster, we again went on manually checking what images goes into these clusters. On checking we found that only one image was present in the cluster.



Silhouette Score: 0.15934951603412628

Davies-Bouldin Score: 3.6421174240769894

Cluster 0(Medium): 25
Cluster 1(High): 75
Cluster 2(Low): 73

We label these clusters as low, medium and high by manual inspection

SLIDESMANIA.COM

Steps

9 Using CNN for Classification

- We attempted to enhance our image features by employing Convolutional Neural Networks (CNN).
- Transfer learning was utilized to load a pre-trained CNN model, with certain layers excluded to focus on detecting edges and corners.
- Features extracted from CNN appeared more abstract compared to previous methods, attributed to the tensor outputs containing numerous numerical values.
- Subsequently, features were extracted using this approach and used for K-Means clustering with three desired clusters.

SLIDESMANIA.COM

Results

9

Using CNN for Classification

- Upon manual inspection, a more balanced distribution of images was observed.
- However, the clustering scores of Silhouette and Davies-Bouldin were lower when utilizing CNN-based features.
- This was primarily due to the difficulty in classifying some images into High, Medium, and Low complexity ranges.

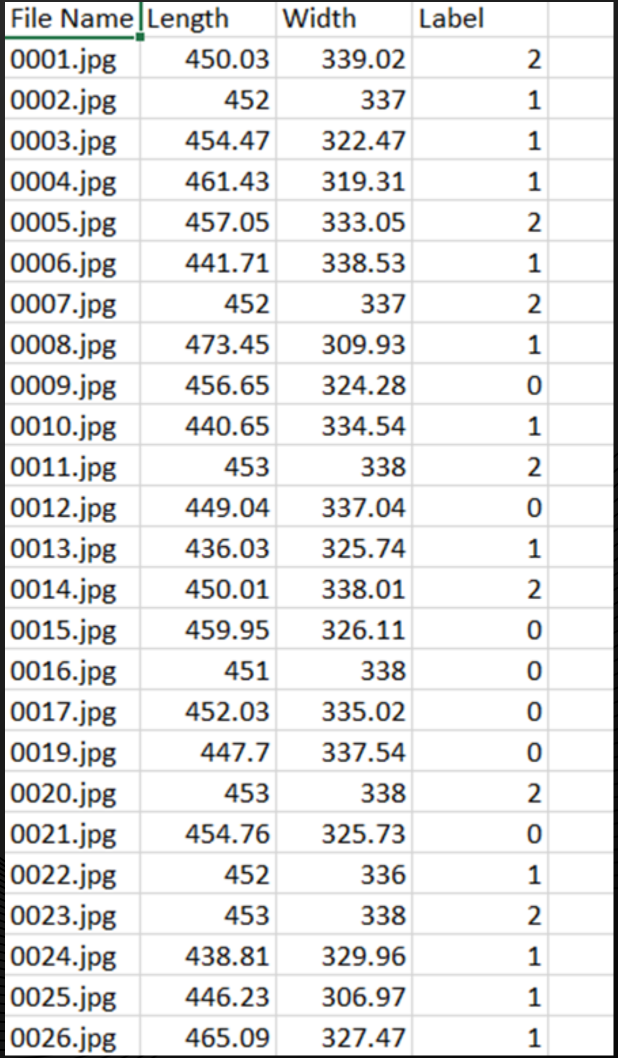
SLIDESMANIA.COM

Steps

10 Tight fitting Box

We incorporated a code to measure the dimensions of a tight-fitting box. Now, when the user provides input such as the box length, box width, and complexity label, the system displays the corresponding images in the next slides.

Since features extracted from CNN are very abstract we have directly created csv file as shown .

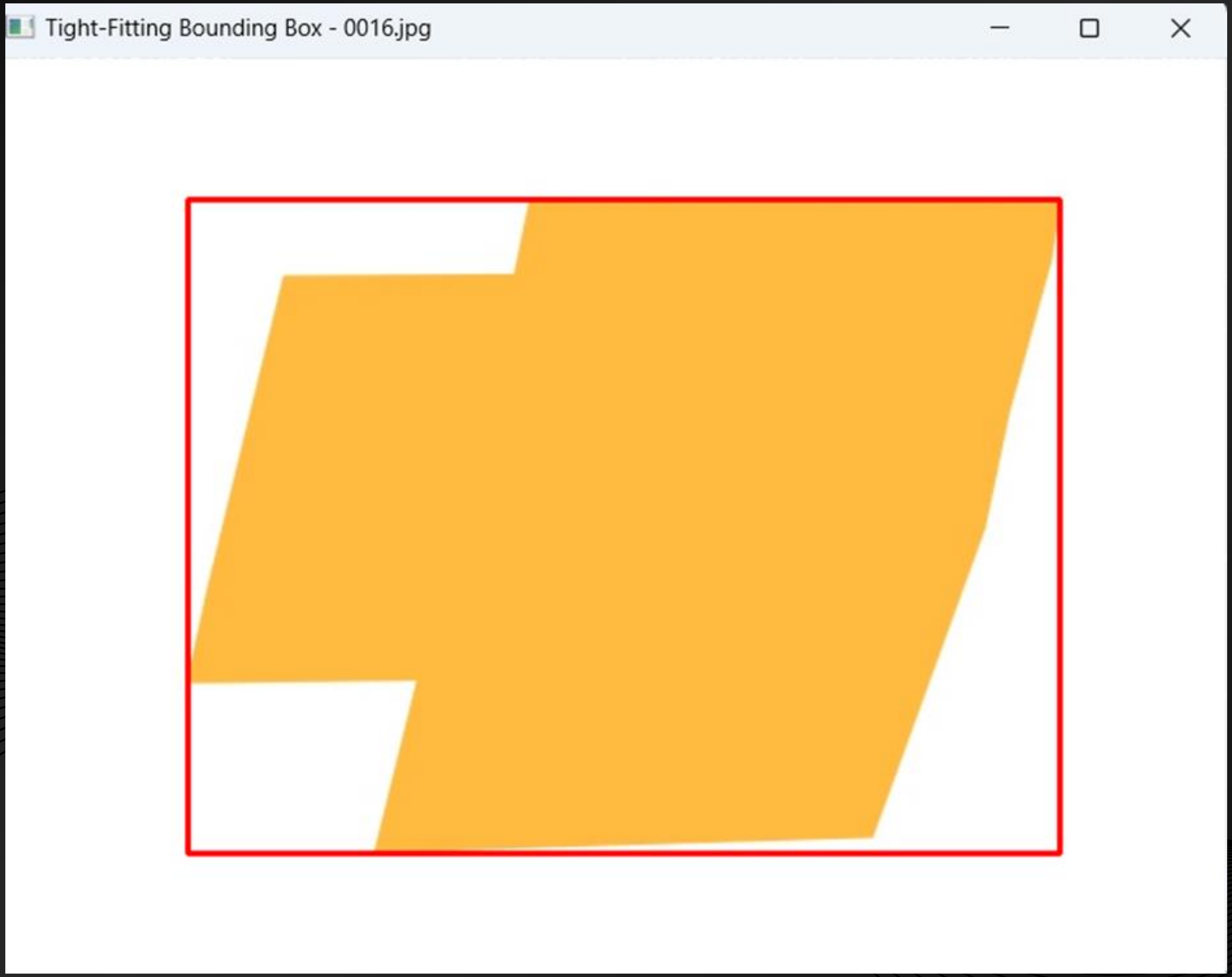


File Name	Length	Width	Label
0001.jpg	450.03	339.02	2
0002.jpg	452	337	1
0003.jpg	454.47	322.47	1
0004.jpg	461.43	319.31	1
0005.jpg	457.05	333.05	2
0006.jpg	441.71	338.53	1
0007.jpg	452	337	2
0008.jpg	473.45	309.93	1
0009.jpg	456.65	324.28	0
0010.jpg	440.65	334.54	1
0011.jpg	453	338	2
0012.jpg	449.04	337.04	0
0013.jpg	436.03	325.74	1
0014.jpg	450.01	338.01	2
0015.jpg	459.95	326.11	0
0016.jpg	451	338	0
0017.jpg	452.03	335.02	0
0019.jpg	447.7	337.54	0
0020.jpg	453	338	2
0021.jpg	454.76	325.73	0
0022.jpg	452	336	1
0023.jpg	453	338	2
0024.jpg	438.81	329.96	1
0025.jpg	446.23	306.97	1
0026.jpg	465.09	327.47	1

SLIDESMANIA.COM

Examples 10

Tight fitting Box



Input:

- Length: 451.0
- Width : 338.0
- Complexity: 0

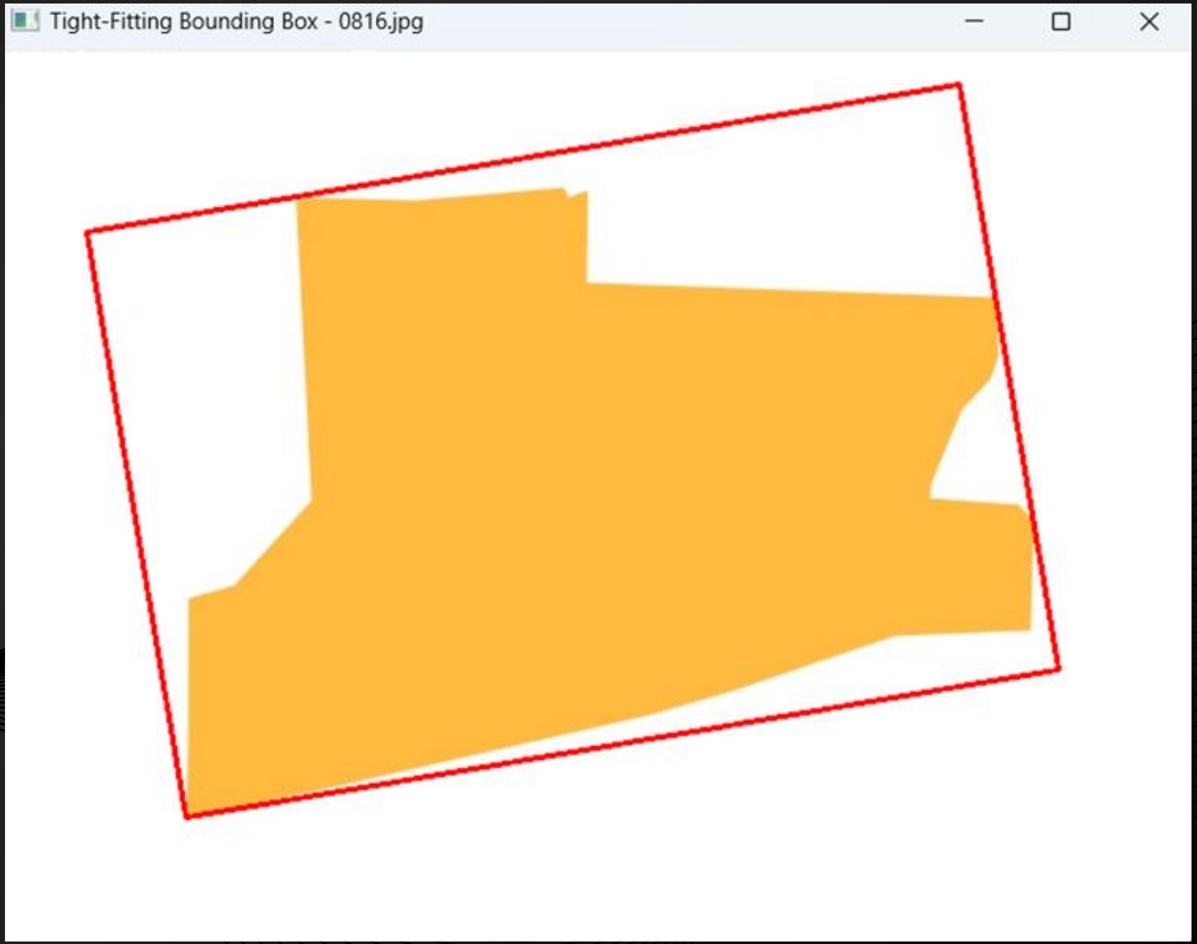
SLIDESMANIA.COM

Examples 10

Tight fitting Box

Input:

- Length: 472.65
- Width : 317.46
- Complexity: 1


Tight-Fitting Bounding Box - 0816.jpg

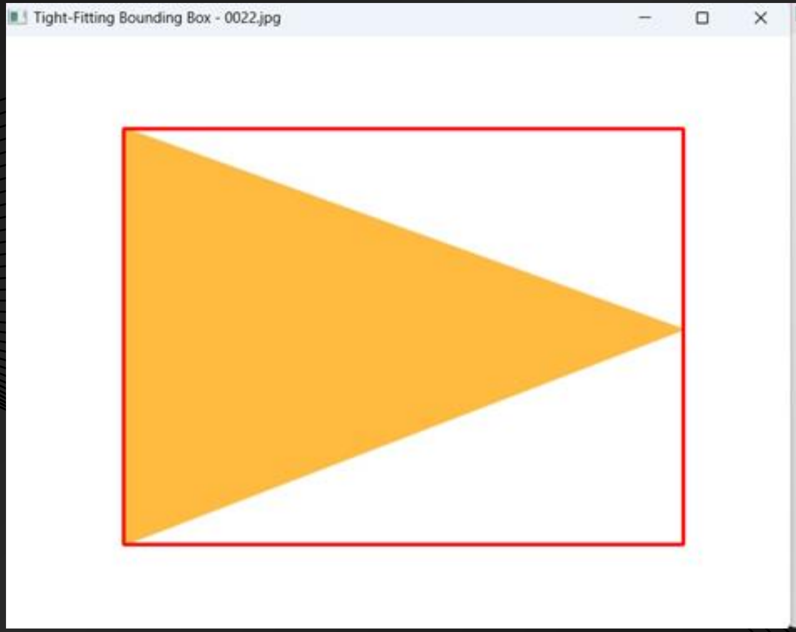
SLIDESMANIA.COM

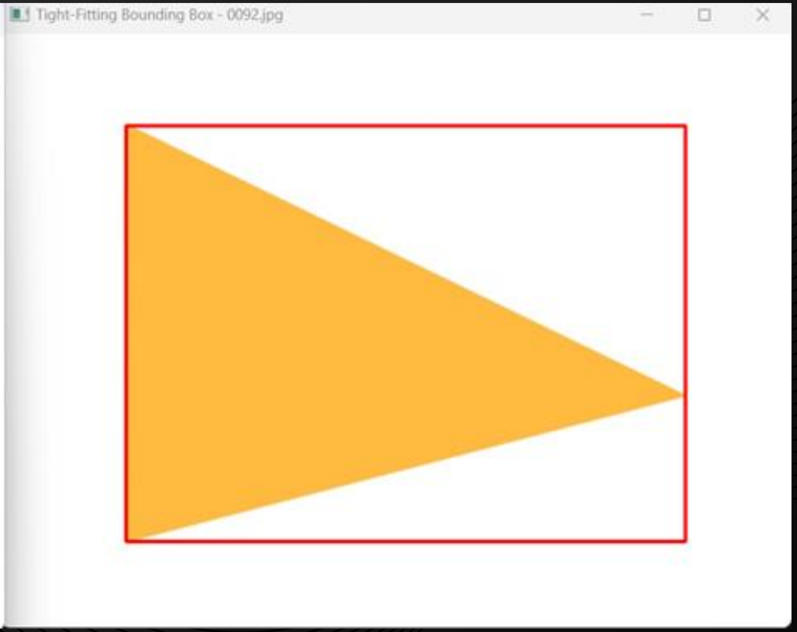
Examples 10

Tight fitting Box

Input:

Length: 452 Width : 336 Complexity: 1





SLIDESMANIA.COM

Summary

Preprocessing Techniques: We started by identifying yellow and white pixels in images, a critical step in our image analysis process. This allowed us to gauge the distribution of these specific colors within our image set.

Edge and Corner Detection: We employed Canny corner detection to pinpoint the corners within the images. The accuracy of this method was validated when the count of these points matched our manual counts.

t-SNE Visualization: To visualize our clustering, we utilized t-SNE. The overlapping of some points indicated that not all clusters were distinct, a valuable insight for our analysis.

Feature Extraction for Clustering: Due to the issues with edge counting, we extracted features such as the 7 Hu Moments and the area and perimeter of the images. These features proved crucial for effective K-Means Clustering.

SLIDESMANIA.COM

Summary

We identified and removed duplicate images using SHA-256 hashing, which not only ensured the accuracy of our analysis but also significantly reduced our image count, influencing our data processing decisions.

After removing duplicates and cleaning our dataset, our K-Means clustering became more representative of image complexity, and while the silhouette and Davies-Bouldin scores indicated quality clustering, but we still conducted manual inspections for verification.

Final Clustering and Labeling: In the final phase, we labeled clusters as low, medium, or high complexity based on manual inspection.

CNN for Feature Extraction: Our team utilized a Convolutional Neural Network (CNN) with transfer learning for feature extraction. This approach yielded more abstract features and improved the results of our manual inspections.

SLIDESMANIA.COM

Important Note

We ran all the code files in Kaggle environment since many of the codes require image processing and using GPU was needed. So we have created all the files and downloaded those locally to check the nature of clusters. So in the zip file we may have some subdirectories

just for representation of codes ran. We were unable to run all the codes locally due to computational constraints and hence our csv files for some models were not created. And since we are creating folders on the basis of these csv files, we are unable to execute some codes with the data files kept the same directory.

However we have included all possible intermediate input data files containing feature information after feature engineering, or output files resulting from analysis or model executions.

We have at last used features which were extracted with CNN since on manual inspection the low, medium and high complexity image were better distributed than the features of Area, Perimeter and Hu Moments. And we have also further used the labels given by features of CNN to take input from user to speed up his layout finding process.

SLIDESMANIA.COM

Program Flow

- Run **preprocessing.ipynb** file to perform all the preprocessing as stated in the slides
- Then run **k-means-1.ipynb** to see what was the behaviour of clustering when we ran it directly on the 1183 images.
- To remove duplicate images and store all the unique images in the separate folder, we use the code for **remove_duplicates.py**
- Then we can see the comparison of two features where we used features by running CNN and by extracting features of each image like Perimeter,Area,Hu moments of contour. It is achieved by running **k-means-2.ipynb**
- To manually inspect what type of images are clustered by using two features ,we first create two csv files named **image_cluster_labels.csv** and **image_features_with_labels_KMeans.csv** . Then we use these two csv files to create subdirectories of clusters using the code written in **Cluster_with_CNN.py** and **Cluster_with_Kmeans.py**
- Now we use code of **tight_fitALL.py** to create features of length and width of tight-fitting box of all images into a csv file.
- We merge the above created csv file with the labels of images as obtained from using features(extracted from CNN) using the code in **csv_merge.py** which then creates the file **merged_csvCNN.csv**; Now we use this CSV file for searching the image, when the user inputs the length,width of tight-fitting box and the complexity of image expected. Then user can see all the possible images with required specifications.

SLIDESMANIA.COM

Intermediate Outputs

- **E7-Images_173** consist of all the subfolders **E7-images(via CNN)** and **E7-images(via k-means)** which then have subfolders representing the 3 clusters as achieved by code described above

- **E7-images_Unique** have all the 173 images after removing all the duplicates

- **E7-images_1183** consist of a subfolder **resized_images** and two csv files named **image_features_with_labels.csv** which represent labels of all the images. Using this csv file we have created 6 subfolders in **resized_images** subfolder. Now the **image_features_with_labels_Cluster_0.csv** is created when we run k-means on **cluster_0** in **resized_images/Cluster_0**. Then we get 100 clusters within cluster_0. We create those 100 subfolders using the csv file of **image_features_with_labels_Cluster_0**.

- **Image_cluster_labels.csv** represents labels corresponding to file names when we use features which were extracted through CNN.

- **image_features_with_labels_KMeans.csv** represents all the features of Area,Perimeter,Hu Moments and label associated with 173 unique images

- **image_features.csv** represents the Length,Width of tight fitting box of all images.

- **merged_csvCNN.csv** is the combined features from two csv files of **image_features.csv** and **Image_cluster_labels.csv**

SLIDESMANIA.COM

Thank you!

SLIDESMANIA.COM