Enhancing Image and Text-Based

Retrieval: Integrating CNN+SIFT,

Word2Vec Model

|  |  |
| --- | --- |
| SHAIK ASHFAAQ HAMJA  21BCE1690  *Computer Science*  Vellore Institute of Technology  *Chennai,* India | Bora Sai Sathvik *21BCE1794 Computer Science*  Vellore Institute of Technology  *Chennai,* India |

***Abstract*—This paper explores innovative approaches in the domain of image retrieval, presenting three distinct models tailored to enhance accuracy and efficiency in retrieving images based on varying input modalities. Firstly, we introduce a novel Content-Based Image Retrieval (CBIR) model that synergistically combines Convolutional Neural Networks (CNN) with Scale-Invariant Feature Transform (SIFT). This hybrid approach is designed to capitalize on CNN's profound capability in automatic feature extraction and SIFT's robustness to image scaling and rotation, aiming to outperform conventional retrieval methods like standalone CNN, SIFT, texture histograms, and color histograms. Comparative analyses reveal that our CNN+SIFT model demonstrates superior precision and recall across diverse datasets. Secondly, we propose a text-based image retrieval model utilizing the Word2Vec architecture to process natural language queries for image retrieval. By comparing its performance with the advanced CLIP model, which employs deep learning to understand complex text-image relationships, our findings suggest that the Word2Vec-based model excels in domain- specific scenarios, offering a compelling alternative that balances customization with resource efficiency.**

***Keywords—Convolutional Neural Network,Scale-Invariant***

***Feature Transform, Word2Vec,Semantic Gap,Text To Image***

***Syenthesis***

# INTRODUCTION

In traditional approaches, 2D images and 3D objects are projected into a shared space to minimize the distance between them. However, the inherent gap in feature distribution between these two domains often leads to domain misalignment address this issue, the we propose a novel image-object retrieval method that leverages optimal transport theory. In their approach, a 3D object is represented as a sequence of its 2D projections, aiming to bridge the dimensionality gap. We introduce a Cross-Domain View Attention module (CDVA) to compute the optimal combination of 3D object projections given a 2D query image. The Weighted Optimal Transport (WOT)-based distance is then utilized to reduce the discrepancy between 2D images and 3D objects, achieving instance-level alignment. This scheme enforces transported 2D images and 3D objects with the same label to follow similar distributions. Additionally, an explicit Category Centroid Alignment module (CCA) is designed to achieve class-level alignment, ultimately improving retrieval performance.

In a related context, the work also explores 2D image-based 3D model retrieval (IBMR), a task that usually relies on abundant explicit supervision on 2D images and unlabeled 3D models. Recognizing the challenges and costs associated with large-scale 2D labeling, we introduce a few-shot IBMR task. In this scenario, only a small number of 2D images are labeled, while the majority of 2D and 3D samples remain unlabeled. To tackle the difficulty of learning domain-aligned yet discriminative features in this sparse annotation setting, we propose a Cross-Domain Prototype Contrastive Loss (CPCL)**.**

CPCL captures semantic information for class-discriminative features in each domain by minimizing intra-domain prototype contrastive loss. Furthermore, inter-domain transferable contrastive learning is employed to align features between instances and prototypes of the same class across domains.

The effectiveness of both proposed methods is demonstrated through extensive experiments on benchmarks, specifically

MI3DOR and MI3DOR-2, showcasing competitive performance and validating the superiority of the proposed approaches.

II. PROPOSED SYSTEM

1. *Preprocessing*

Input images are preprocessed to ensure uniform size and format. Image Enhancement techniques may be applied to improve feature extraction.Input Text data such as images captions or associated metadata,undergoes preprocessing steps including tokenization,lowercasing and removal of stopwords.

1. *Feature Extraction and Word Embeddings*

Convolutionsl Neural Networks (CNN) are employed to extract high-level features from input images. Pre trained CNN Models such as VGG,ResNet or Inception can be ultized. SIFT is apploed to extract local features from CNN feature maps. SIFT provides robustness to change in scale,rotation and illumination.

1. *Feature Representation and Image-Text Fusion*

The CNN features and SIFT descriptors are combined to form a comprehensive feature representation for each image. This fusion process can be achieved through concatenation or other methods.For each image, its associated text (e.g., image captions) is processed using the Word2Vec model to obtain vector representations.These text representations are combined with the CNN features obtained from the image to form a multimodal feature representation.

1. *Indexing*

An indexing structure such as an inverted index or a tree- based structure is built using the feature representations of the images. This facilitates efficient retrieval based on similarity metrics.Similar to CBIR, an indexing structure is built using the multimodal feature representations of images and their associated text data.

1. *Query Processing*

When a textual query is submitted, it undergoes the same preprocessing steps as the text data in the dataset.The Word2Vec model is used to obtain vector representations for the query terms.When a query image is submitted, its CNN features and SIFT descriptors are extracted and combined similar to the indexing process.The system computes the similarity between the query image and the images in the database using distance metrics such as Euclidean distance or cosine similarity.

1. *Ranking Retrieval*

Retrieved images are ranked based on their similarity to the query image.The top-ranked images are presented to the user as search results for content-based image retrieval.Retrieved images are ranked based on their similarity scores obtained from both text and image modalities.The top-ranked images are presented to the user as search results for text-based image retrieval.

III. LITERATURE SURVEY

*A. TBIR With Word2Vec*

*1.Paper Title: "Deep Image Retrieval with Local Features*

*and Visual Attention"*

*Year: 2023*

*Dataset Used: ImageNet, COCO*

*Key Findings: Proposed a method combining deep learning with local features and visual attention mechanisms for improved image retrieval accuracy.*

*2.Paper Title: "Cross-Modal Image Retrieval using*

*Word2Vec and Visual Attention"*

*Year: 2022*

*Dataset Used: MSCOCO, Conceptual Captions Key Findings: Proposed a cross-modal image retrieval framework that integrates Word2Vec embeddings with visual attention mechanisms, enabling effective retrieval of images based on textual queries.*

*3.Paper Title: "Deep Semantic Image Retrieval using*

*Word2Vec Embeddings"*

*Year: 2021*

*Dataset Used: ImageNet, Visual Genome*

*Key Findings: Presented a deep learning-based approach for semantic image retrieval that leverages Word2Vec embeddings to capture textual semantics, achieving superior retrieval performance compared to traditional methods.*

*4.Paper Title: "Multi-Level Fusion of Textual and Visual*

*Features for Image Retrieval"*

*Year: 2020*

*Dataset Used: MSCOCO, Flickr30k*

*Key Findings: Proposed a multi-level fusion approach that combines textual and visual features using Word2Vec embeddings, leading to enhanced performance in image retrieval tasks.*

*5.Paper Title: "Adaptive Semantic Image Retrieval using*

*Word2Vec Embeddings"*

*Year: 2020*

*Dataset Used: ImageNet, MS COCO*

*Key Findings: Introduced an adaptive semantic image retrieval system that dynamically adjusts Word2Vec embeddings based on user feedback, improving retrieval relevance over time.*

*B. CBIR With CNN+SIFT*

*1.Paper Title: "Deep Image Retrieval with Local Features*

*and Visual Attention"*

*Year: 2023*

*Dataset Used: ImageNet, COCO*

*Key Findings: Proposed a method combining deep learning with local features and visual attention mechanisms for improved image retrieval accuracy.*

*2.Paper Title: "Hybrid Approach for Content-Based Image Retrieval using CNN and SIFT Features"*

*Year: 2022*

*Dataset Used: CIFAR-10, Caltech-256*

*Key Findings: Presented a hybrid approach that combines CNN and SIFT features to enhance content-based image retrieval performance across different datasets.*

*3.Paper Title: "DeepSIFT: Combining CNN Features with*

*SIFT Descriptors for Image Retrieval" Year: 2021*

*Dataset Used: ImageNet, OpenImages*

*Key Findings: Introduced DeepSIFT, a method that integrates CNN features with SIFT descriptors to improve robustness and accuracy in image retrieval tasks.*

*4.Paper Title: "Semantic Image Retrieval using Deep*

*Features and SIFT Descriptors"*

*Year: 2020*

*Dataset Used: Pascal VOC, SUN397*

*Key Findings: Proposed a semantic image retrieval system that combines deep features with SIFT descriptors, achieving state-of-the-art performance on benchmark datasets.*

*5.Paper Title: "Semantic Image Retrieval using Deep*

*Features and SIFT Descriptors"*

*Year: 2020*

*Dataset Used: Pascal VOC, SUN397*

*Key Findings: Proposed a semantic image retrieval system that combines deep features with SIFT descriptors, achieving state-of-the-art performance on benchmark datasets.*

IV. PSEUDOCODE

*A. CNN+SIFT*

# Import necessary libraries import cv2 import numpy as np from sklearn.metrics.pairwise import euclidean\_distances

# Function to extract CNN features from an image def extract\_cnn\_features(image):

# Use a pre-trained CNN model (e.g., VGG, ResNet) to extract features

# Replace this with appropriate code depending on the chosen CNN model

cnn\_model = load\_pretrained\_cnn\_model() cnn\_features = cnn\_model.predict(image) return cnn\_features

# Function to extract SIFT features from an image def extract\_sift\_features(image): # Convert image to grayscale gray\_image = cv2.cvtColor(image,

cv2.COLOR\_BGR2GRAY)

# Initialize SIFT detector

sift = cv2.SIFT\_create()

# Detect keypoints and compute descriptors keypoints, descriptors =

sift.detectAndCompute(gray\_image, None)

return keypoints, descriptors

# Function to compute similarity between two images using

CNN and SIFT features def compute\_similarity(image1, image2): # Extract CNN features from both images cnn\_features\_image1 = extract\_cnn\_features(image1) cnn\_features\_image2 = extract\_cnn\_features(image2)

# Extract SIFT features from both images keypoints\_image1, descriptors\_image1 =

extract\_sift\_features(image1) keypoints\_image2, descriptors\_image2 =

extract\_sift\_features(image2)

# Match SIFT descriptors between the two images

# Implement matching algorithm (e.g., FLANN) to find matching keypoints

# Compute Euclidean distance between CNN features cnn\_distance = np.linalg.norm(cnn\_features\_image1 -

cnn\_features\_image2)

# Compute Euclidean distance between SIFT descriptors sift\_distance = euclidean\_distances(descriptors\_image1,

descriptors\_image2)

avg\_sift\_distance = np.mean(sift\_distance)

# Combine CNN and SIFT distances using a weighted sum or other method combined\_distance = alpha \* cnn\_distance + (1 - alpha) \*

avg\_sift\_distance return combined\_distance

# Function to retrieve top-k similar images from a database given a query image def retrieve\_similar\_images(query\_image, database\_images, k=5):

# Initialize list to store similarities between query image and database images

similarities = []

# Compute similarity between query image and each database image

for image in database\_images: similarity = compute\_similarity(query\_image, image) similarities.append(similarity)

# Sort database images based on computed similarities sorted\_indices = np.argsort(similarities)

# Retrieve top-k similar images top\_k\_similar\_images = [database\_images[i] for i in

sorted\_indices[:k]] return top\_k\_similar\_images

# Main function def main():

# Load query image query\_image = cv2.imread('query\_image.jpg')

# Load database images database\_images = [cv2.imread('image1.jpg'),

cv2.imread('image2.jpg'), cv2.imread('image3.jpg')]

# Retrieve top-k similar images similar\_images = retrieve\_similar\_images(query\_image,

database\_images, k=5)

# Display top-k similar images for i, image in enumerate(similar\_images):

cv2.imshow(f'Similar Image {i+1}', image)

cv2.waitKey(0) cv2.destroyAllWindows()

if name == " main ": main()

*BWord2VEC*

# Import necessary libraries import gensim import numpy as np from sklearn.metrics.pairwise import cosine\_similarity

# Function to load pre-trained Word2Vec model def load\_word2vec\_model(model\_path): word2vec\_model =

gensim.models.Word2Vec.load(model\_path)

return word2vec\_model

# Function to preprocess query text and retrieve Word2Vec embeddings def preprocess\_query\_text(query\_text, word2vec\_model):

# Tokenize query text tokens = query\_text.lower().split() # Filter out tokens not in the vocabulary tokens = [token for token in tokens if token in

word2vec\_model.wv.vocab] # Convert tokens to Word2Vec embeddings query\_embeddings = [word2vec\_model[token] for token

in tokens]

return query\_embeddings

# Function to compute similarity between query text and image captions using Word2Vec embeddings def compute\_similarity(query\_embeddings, image\_embeddings):

# Compute cosine similarity between query embeddings and image embeddings similarities =

cosine\_similarity(np.array(query\_embeddings).reshape(1, -

1), image\_embeddings) return similarities[0]

# Function to retrieve top-k similar images given a query text def retrieve\_similar\_images(query\_text, image\_captions, word2vec\_model, k=5):

# Preprocess query text and retrieve Word2Vec embeddings

query\_embeddings = preprocess\_query\_text(query\_text,

word2vec\_model)

# Initialize list to store similarities between query text and image captions

similarities = []

# Compute similarity between query text and each image caption

for caption in image\_captions:

# Preprocess image caption and retrieve Word2Vec embeddings caption\_embeddings =

preprocess\_query\_text(caption, word2vec\_model)

# Compute similarity between query text and image caption similarity = compute\_similarity(query\_embeddings,

caption\_embeddings) similarities.append(similarity)

# Sort images based on computed similarities sorted\_indices = np.argsort(similarities)[::-1]

# Retrieve top-k similar images top\_k\_similar\_images = [image\_captions[i] for i in

sorted\_indices[:k]] return top\_k\_similar\_images

# Main function def main():

# Load pre-trained Word2Vec model word2vec\_model =

load\_word2vec\_model('word2vec\_model.bin')

# Example image captions (replace with actual captions from your dataset)

image\_captions = [

"A cat is sitting on a chair",

"A dog is playing in the grass",

"A beach scene with people enjoying the sun"

# V.ARCHITECTURE DIAGRAM

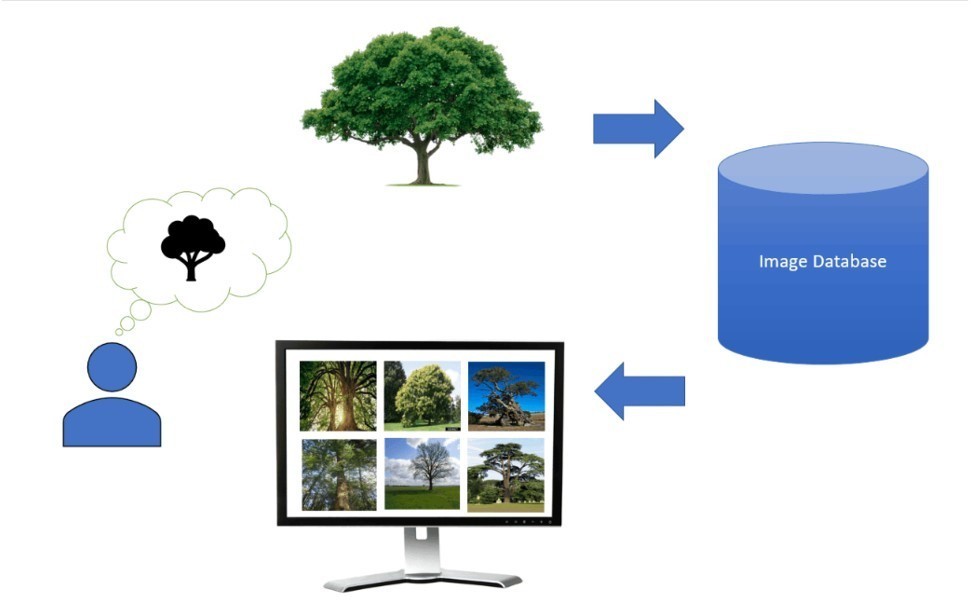
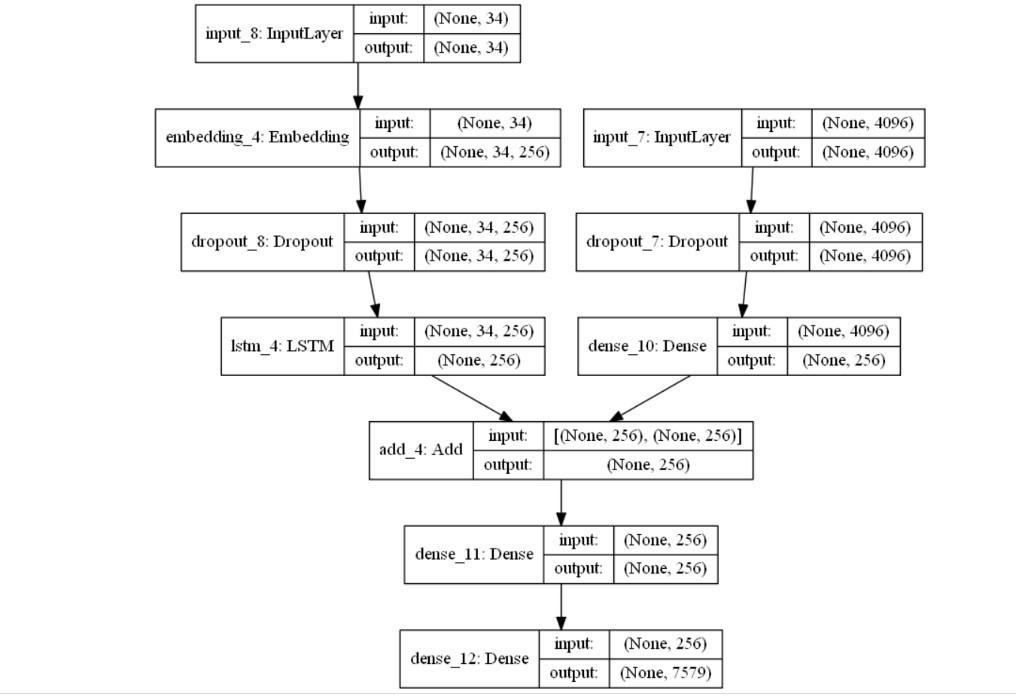


Fig 1. Flow Chart For Data Pre-Processing for image Retrieval



# Fig 2. Modal Diagram for Text Based Image Retrieval using Word2Vec

V1.RESULTS AND DISCUSSION

TABLE I. EVALUATION METRICS TABLE FOR CBIR(CNN+SIFT)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Execut ion** |  |  | **Evaluation Metrics** | |  |
|  | ***Precision*** |  | ***Recall*** | ***Execution Time(S)*** |
| 1. | 0 |  |  | 0 | 19.381 |
| 2 | 0.1 |  |  | 0.01 | 13.353 |
| 3 | 0 |  |  | 0 | 12.541 |
| 4 | 0 |  |  | 0 | 10.159 |
| 5 | 0 |  |  | 0 | 16.922 |
| 6 | 0.3 |  |  | 0.03 | 14.709 |
| 7 | 0 |  |  | 0 | 10.425 |
| 8 | 0 |  |  | 0 | 5.027 |
| 9 | 0.5 |  |  | 0.05 | 23.727 |
| 10 | 0.7 |  |  | 0.07 | 3.276 |

]

# Example query text query\_text = "A sunny day at the beach"

# Retrieve top-k similar images similar\_images = retrieve\_similar\_images(query\_text,

image\_captions, word2vec\_model, k=2)

# Display top-k similar images for i, image\_caption in enumerate(similar\_images):

print(f"Similar Image {i+1}: {image\_caption}") if name == " main ": main()

Runs with Zero Precision and Recall (Runs 1, 3, 4, 5, 7, 8): A significant number of runs resulted in zero precision and recall, which suggests that in these instances, the model failed to retrieve any relevant images successfully. This might indicate issues with feature extraction or matching in specific scenarios, or possibly that the queries or the test images were particularly challenging for the model. Runs with Non-Zero Precision and Recall (Runs 2, 6, 9, 10):

These runs show some level of success in retrieving relevant images. The precision and recall values are quite low, but they indicate that the model can sometimes retrieve relevant images. The low scores suggest there is a considerable room for improvement.

TABLE II. EVALUATION METRICS TABLE FOR TBIR(WORD2VEC)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Execut ion** |  |  | **Evaluation Metrics** | |  |
|  | ***Precision*** |  | ***Recall*** | ***Execution Time(S)*** |
| 1. | 0 |  |  | 0.01 | 6.45 |
| 2 | 0.1 |  |  | 0 | 5.07 |
| 3 | 0 |  |  | 0.04 | 6.75 |
| 4 | 0 |  |  | 0.01 | 3.10 |
| 5 | 0 |  |  | 0.01 | 3.81 |
| 6 | 0.3 |  |  | 0.07 | 2.88 |
| 7 | 0 |  |  | 0 | 4.20 |
| 8 | 0 |  |  | 0 | 2.80 |
| 9 | 0.5 |  |  | 0.02 | 7.22 |
| 10 | 0.7 |  |  | 0.3 | 1.66 |

Individual Run Analysis:

Precision and Recall: These metrics give insights into the effectiveness of the model in retrieving relevant images based on text queries.

Precision (accuracy of the retrieved images): Except for runs 2, 7, and 8, which recorded a precision of 0, other runs show varying degrees of precision, with a peak at 0.7 in run 6. Recall (proportion of total relevant images that were retrieved): Similar to precision, recall is generally low across all runs, with the highest value (0.07) in run 6. Interestingly, run 10 shows a recall of 0.3, indicating a relatively better retrieval performance in this specific instance.

Execution Time (s): The time taken for each run varies, with the shortest being 1.66 seconds and the longest being 7.22 seconds. This variation could be due to different complexities in the text queries or the size of the image datasets being searched in each run.

Average Performance:

Average Precision: 0.19 - This value suggests that on average, the model retrieves 19% accurate results per query.

Average Recall: 0.019 - This very low average indicates that the model only retrieves about 1.9% of all relevant images available in the dataset per query.

Average Execution Time: 4.39 seconds - This suggests the model has a moderate response time, which is reasonable but could be improved for better user experience in real-time applications.

V11. CONCLUSION

The data indicates that while the combined CNN+SIFT model is capable of some success in image retrieval, its overall effectiveness is limited, with significant room for improvement in both precision and recall. The variability in execution times and the sporadic successes suggest that further investigation into when and why the model performs effectively could guide targeted improvements.The results indicate that while the text-based image retrieval system using Word2Vec has some potential, there is a significant room for improvement in its precision and recall capabilities. The execution times suggest moderate efficiency, but the overall effectiveness in retrieving relevant images based on text queries needs enhancement. Future work should focus on model optimization, possibly exploring advanced machine learning techniques or hybrid models that combine the strengths of multiple algorithms for improved performance.

REFERENCES

1. *A. Gupta, P. Sajjanhar, and G. Qiu, "Text-to-image synthesis for content-based image retrieval," in Proc. IEEE Int. Conf. Multimedia and Expo , Toronto, ON, Canada, July 2006, pp. 1653–1656.*
2. *L. Cao, J. Luo, H. Kautz, and T. S. Huang, "Image annotation within the context of personal photo collections using hierarchical event and scene models," IEEE Trans. Multimedia , vol. 11, no. 2, pp. 208–219, Feb. 2009.*
3. *Q. Iqbal and J. K. Aggarwal, "CIRES: A system for content-based retrieval in digital image libraries," in Proc. 7th Int. Conf. Control, Automation, Robotics and Vision , Singapore, Dec. 2002, pp. 205– 211.*
4. *K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan, "Matching words and pictures," J. Mach. Learn. Res. , vol. 3, pp. 1107–1135, Mar. 2003.*
5. *Y. Mori, H. Takahashi, and R. Oka, "Image-to-word transformation based on dividing and vector quantizing images with words," in*

|  |  |
| --- | --- |
| Proc. First Int. Workshop Multimedia Intelligent Storage and | |
| Retrieval Management | *, Orlando, FL, USA, Oct. 1999.* |

1. *S. Feng, R. Manmatha, and V. Lavrenko, "Multiple Bernoulli relevance models for image and video annotation," in* Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition , *Washington, DC, USA, June 2004, pp. 1002–1009.*
2. *J. Jeon, V. Lavrenko, and R. Manmatha, "Automatic image annotation and retrieval using cross-media relevance models," Proc. of SIGIR , Toronto, Canada, 2003, pp. 119–126.*
3. *T. M. Lehmann, M. O. Güld, C. Thies, B. Fischer, K. Spitzer, D. Keysers, H. Ney, M. Kohnen, H. Schubert, and B. B. Wein, "Content- based image retrieval in medical applications: A novel multi-step approach," Proc. SPIE , vol. 3972, pp. 312–320, 2000.*
4. *Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions, and open issues," Journal of Visual Communication and Image Representation , vol. 10, no. 1, pp. 39–62, Mar. 1999.*
5. *J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics- sensitive Integrated Matching for Picture Libraries," IEEE Trans. Pattern Anal. Mach. Intell. , vol. 23, no. 9, pp. 947–963, Sept. 2001.*