# Project #3 Feature Extraction and Object Retrieval Report

# Group BJL: Boying Li, Lianhe Chu, Jie Zhong

Project description:

Students will create a pipeline to retrieve a specific object by extracting and matching features, determining similarity to an anchor image. The environment will contain multiple similar objects.

Requirements

• Implement a feature extraction and matching algorithm (e.g., SIFT, SURF, or deep learningbased methods).

• Calculate a similarity score between the target object and anchor image.

• Implement an object retrieval system capable of identifying the correct object among a set of visually similar objects.

Provided material

• Dataset: Custom dataset with similar objects.

----------------------------------------------------------------------------------------------

Project output and report:

# 1. Introduction

This project aims to implement an ‘Object Retrieval System’ capable of identifying specific objects within an environment containing multiple visually similar objects. By using computer vision techniques such as feature extraction, matching, and depth image analysis, the system accurately retrieves objects from a dataset of RGB and depth images. The objective is to find objects that closely match predefined anchor images based on similarity scores.

Project Objectives:

• Implement a feature extraction and matching algorithm (e.g., SIFT, SURF, or deep learningbased methods).

• Calculate a similarity score between the target object and anchor image.

• Implement an object retrieval system capable of identifying the correct object among a set of visually similar objects.

# 2. Dataset

The provided test dataset consists of RGB and depth images stored in separate folders. The RGB images capture color information, while the depth images contain distance information for each pixel. Additionally, camera calibration files with timestamp data are provided, which allows for synchronization of RGB and depth image pairs. The anchor images are also stored separately and represent the specific objects, in this case, fire\_extinguisher images, we aim to retrieve from the test dataset.

# 3. Challenges

We encountered several challenges with the project.

* Unsynchrionized RGB and depth image.
* Anchor image too big and type conversion to png.
* Camera information exploration and usage for project.
* Feature extraction method , SIFT, ORB, and etc.
* Similarity score setting

# 4. Approach

We have implement code to overcome the challenges and the code for the project involves the following main components:

* Data Synchronization and Preparation
* Feature Extraction and Matching
* Similarity Score Calculation
* Object Retrieval
* Visualization of Results

## 4.1 Data Synchronization and Preparation

The anchored image are converted from .heic to .png format to perform further analysis.

Since the RGB and depth images are captured with slight time differences, they must be synchronized. The function `synchronize\_images()` pairs RGB and depth images by finding the closest timestamps for each RGB image in the depth timestamps.

Code Implementation

#--------------------sync rgb and depth pairs-------------------------

rgb\_timestamps = timestampAll(CAMERAINFO\_RGB\_PATH)

depth\_timestamps = timestampAll(CAMERAINFO\_depth\_PATH)

rgb\_images = [cv2.imread(os.path.join(RAW\_RGB\_PATH, file)) for file in os.listdir(RAW\_RGB\_PATH)]

depth\_images = [cv2.imread(os.path.join(RAW\_depth\_PATH, file), cv2.IMREAD\_ANYDEPTH) for file in os.listdir(RAW\_depth\_PATH)]

# Synchronize images

synchronized\_images = synchronize\_images(rgb\_timestamps, depth\_timestamps, rgb\_images, depth\_images)

## 4.2 Feature Extraction and Matching

Two types of features are extracted from the images, we have extract SIFT keypoints and descriptors for feature matching.

Similarity Scoring:

* Color-based features: Used to match regions of the RGB images with the anchor images.

The function `segment\_fire\_extinguisher()` segments specific red regions, which represent the target object the red fire extinguisher. After segmentation, the SIFT (Scale-Invariant Feature Transform) algorithm is used to extract keypoints and descriptors for feature matching.

def segment\_fire\_extinguisher(image):

    hsv = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

    lower\_red1 = np.array([0, 50, 50])

    upper\_red1 = np.array([10, 255, 255])

    lower\_red2 = np.array([170, 50, 50])

    upper\_red2 = np.array([180, 255, 255])

    mask1 = cv2.inRange(hsv, lower\_red1, upper\_red1)

    mask2 = cv2.inRange(hsv, lower\_red2, upper\_red2)

    mask = mask1 | mask2

    segmented\_image = cv2.bitwise\_and(image, image, mask=mask)

return segmented\_image, mask

def segmented\_match\_score(anchor, rgb):

    score = 0

    rgb\_seg, \_ = segment\_fire\_extinguisher(rgb)

    anchor\_seg, \_ = segment\_fire\_extinguisher(anchor)

    anchor\_seg\_kp, anchor\_seg\_des = extract\_sift\_features(anchor\_seg)

    rgb\_seg\_kp, rgb\_seg\_des = extract\_sift\_features(rgb\_seg)

    if rgb\_seg\_kp:

        matches,score = match\_features(anchor\_seg\_des, rgb\_seg\_des)

        print(f"Match Score of red extinguisher: {score}")

    return score

* Depth-based edge features: Used to extract structural similarities from depth images.

The `depth\_match\_score()` function extracts edges from depth images using Gaussian blurring and Canny edge detection. SIFT descriptors are then used to extract features from these edges.

def depth\_match\_score(anchor, depth):

    score = 0

    blurred\_depth = cv2.GaussianBlur(np.array(depth),(5,5),5)

    depth\_edge = edgedepth(blurred\_depth)

    blurred\_anchor = cv2.GaussianBlur(np.array(anchor),(5,5),5)

    anchor\_edge = cv2.Canny(blurred\_anchor, threshold1=50, threshold2=150)

    anchor\_edge\_kp, anchor\_edge\_des = extract\_sift\_features(anchor\_edge)

    depth\_edge\_kp, depth\_edge\_des = extract\_sift\_features(depth\_edge)

    if depth\_edge\_kp:

        matches,score = match\_features(anchor\_edge\_des, depth\_edge\_des)

        print(f"Match Score of depth image: {score}")

    return score

## 4.3 Similarity Score Calculation

The similarity score calculation consists of two primary scores in our approach code is included above:

* RGB Segmentation Similarity: Measures similarity in segmented red regions of the RGB image, which represents the anchor object.
* Depth Edge Similarity: Measures similarity in the edges detected in depth images.

A weighted combination of these scores is used to calculate a final similarity score:

def retrieve\_objects(anchor, rgb, depth, weight=0.1):

    # combine depth similarity and rgb red area similarity with adjustable weight

    depth\_score = depth\_match\_score(anchor, depth)

    red\_score = segmented\_match\_score(anchor, rgb)

    score\_sum = depth\_score + weight\*red\_score

    return score\_sum

## 4.4 Object Retrieval

The function `retrieve\_objects()` shown above combines the RGB and depth similarity scores, which allows the system to rank objects based on similarity to the anchor image. This function returns a score indicating the likelihood of a match.

## 4.5 Visualization of Results

The function `visualize\_score()` displays a bar plot of similarity scores for each image compared with each of the anchor image. This visualization helps to identify images with the highest match scores and validate the system's retrieval performance.

def visualize\_score(scores, anchor\_name):

    # visulaize score bar plot

    x\_tick = range(1, len(scores) + 1)

    # Create a DataFrame

    data = pd.DataFrame({'Image': x\_tick, 'Score': scores})

    # Create the bar plot using Seaborn

    plt.figure(figsize=(10, 6))

    sns.barplot(x='Image', y='Score', data=data)

    # Customizing the plot

    plt.title(f"Similarity Scores of anchor{anchor\_name}")

    plt.xlabel("Image")

    plt.ylabel("Similarity Score (Match Ratio)")

    plt.xticks(rotation=45, fontsize=5)

    # Adjust layout to prevent overlap

    plt.tight\_layout()

    plt.show()

# 5. Code Summary

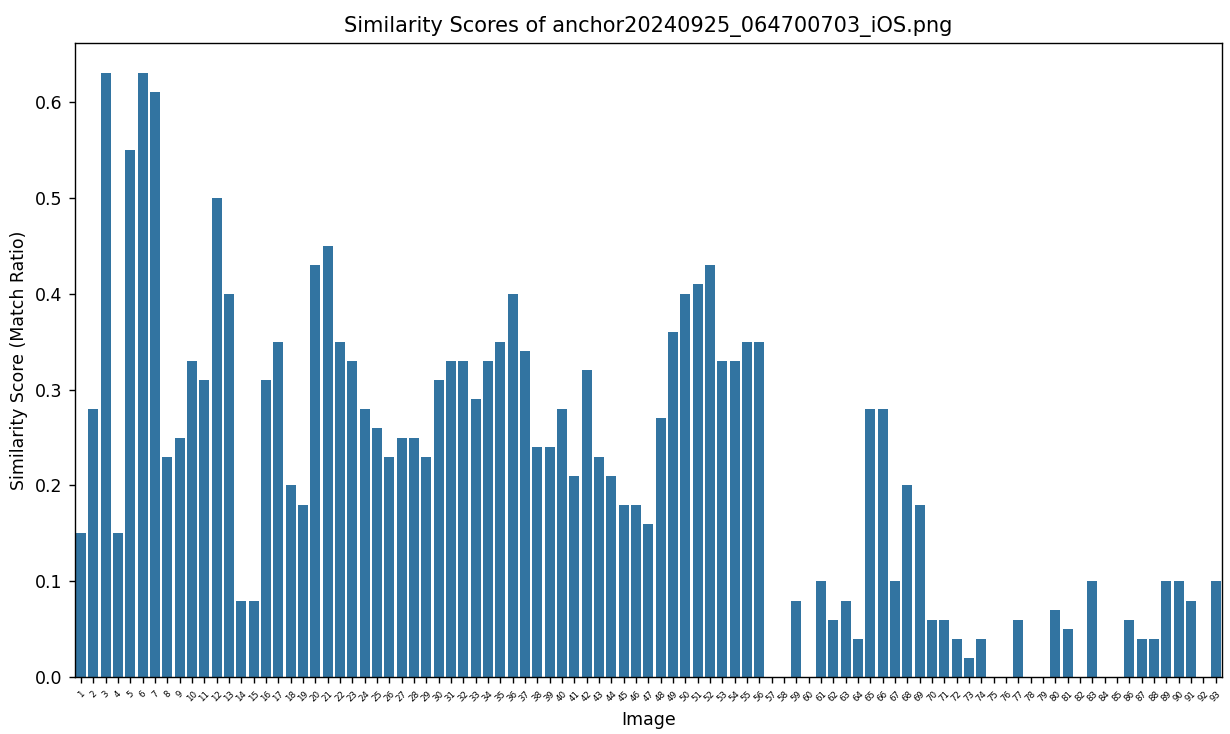
Here is a summary of our implementation of major code functions for the object retrieval pipeline in the project:

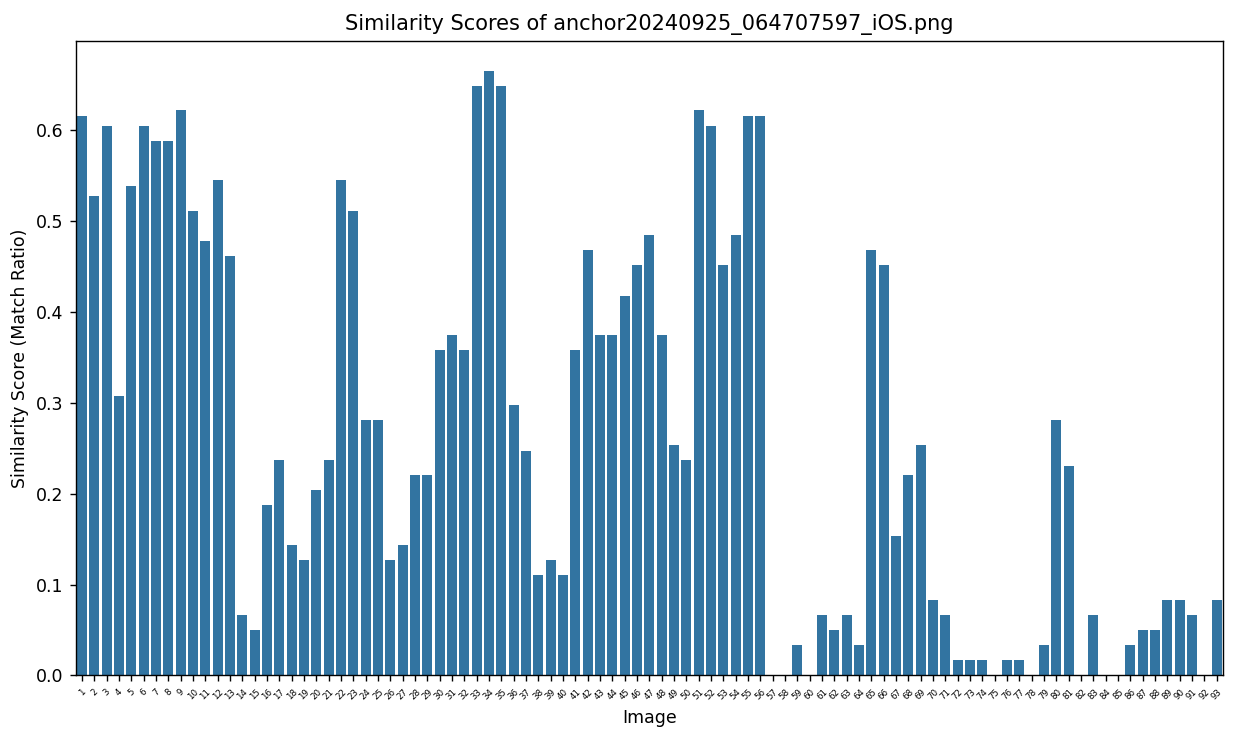
* Data Synchronization:
  + `read\_camera\_info()` extracts camera calibration and timestamp data.
  + `synchronize\_images()` pairs RGB and depth images based on closest timestamp matches.
* Feature Extraction and Segmentation:
  + `segment\_fire\_extinguisher()` segments the object of interest (in our case, a red fire extinguisher) from the RGB images.
  + `extract\_sift\_features()` extracts SIFT keypoints and descriptors for feature matching.
* Similarity Scoring:
  + `segmented\_match\_score()` calculates the match score for RGB images by comparing segmented features.
  + `depth\_match\_score()` calculates the match score for depth images by comparing edge features.
* Object Retrieval and Visualization:
  + `retrieve\_objects()` calculates the overall similarity score by combining RGB and depth scores.
  + `visualize\_score()` displays the similarity scores for visual analysis.

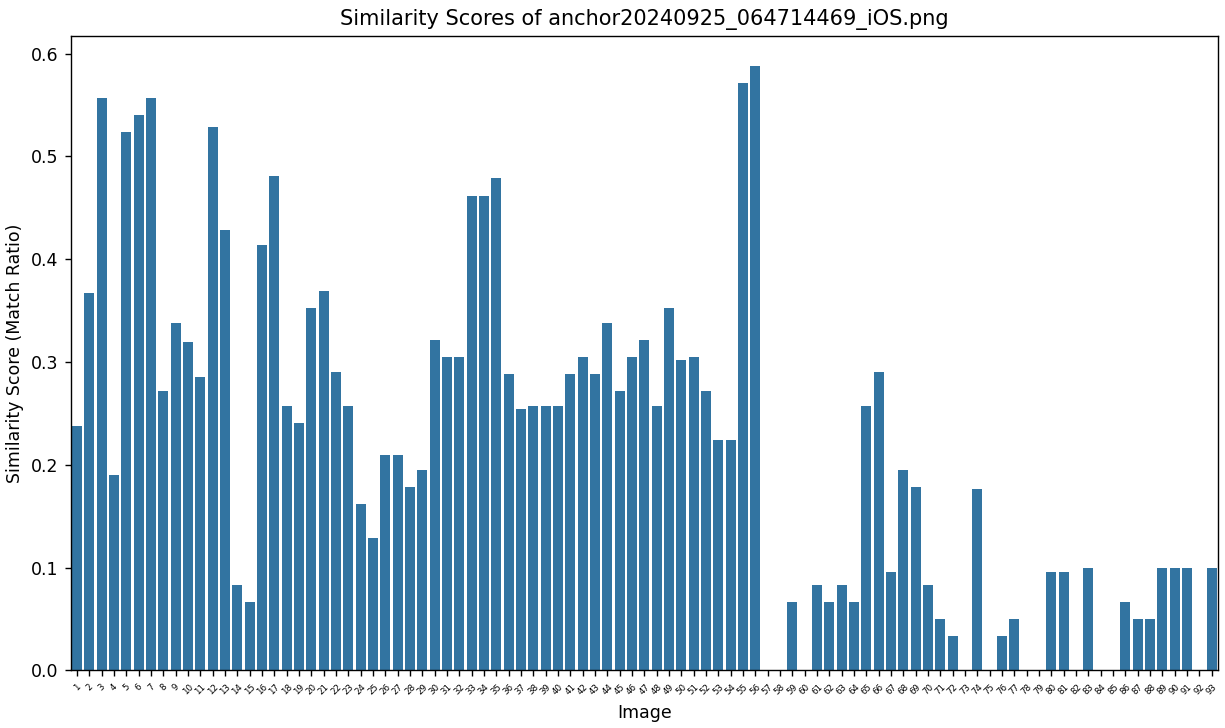
# 6. Results

Our approach for the project calculates a similarity score for each image in the dataset based on its visual resemblance to each of the provided anchor images. The results are visualized in bar plots, where higher scores indicate a closer match to the anchor image. This setup allows us to quickly identify images that most closely match the target object in environments with visually similar items.

Detailed similarity score for all images (93 in total for ...Project\_3\raw\test\camera\_color\_image\_raw) plot for each anchor images (3 in total for …\Project\_3\Project\_3\Anchor\_converted) are shown below:







# 7. Conclusion and Discussion

Our approach for the object retrieval project successfully implements a robust object retrieval pipeline that combines RGB segmentation and depth edge analysis for object similarity detection. We have reached combined similarity score for each rgb image. We could see some of them reach a score over 0.6 while some left empty meaning there is no fire extinguisher detected in the raw rgb image. The combination of RGB and depth features enhances the system’s ability to distinguish between similar objects. In addition, we also demonstrate the effectiveness of SIFT-based feature matching for high-resolution images and provides a foundation for further improvements, such as incorporating deep learning models for feature extraction.