**Course Project**

CP467: Image Processing & Recognition

Professor: Dr. Zia Ud Din

Due Date: Saturday December 2, 2023. 11:59 p.m.

Chandler Mayberry 190688910

Samson Goodenough 190723380

Table of Contents

[Introduction 3](#_Toc152096409)

# Introduction

All the code for this assignment may be run using main.ipynb file in the source code folder. Running this file will perform the operations and write the resulting images to the *Detected\_Objects*, *Keypoints*, and *Matches* folder as .png files. Note: no results are shown within the Jupyter notebook, matplotlib and cv2.imshow was used for testing only.

|  |
| --- |
| Report Includes:   * A small literature review that discusses past and current methods on how to perform tasks (1) and (2) and discuss the advantages and disadvantages of each approach. * The methods and algorithms you have used for feature detection, matching and image stitching, and why did you select those? * A table of results for task (1). * The challenges faces and how they were addressed. * A section for possible improvements or future work. * \*\*\*\*Should not be more than 5 pages.\*\*\*\* * \*\*\*\* The most important component of the report is your table of results.\*\*\*\* |

# Past and Current Methods:

**-Past and current methods for object recognition in a scene:**

Notes:

-Edge detection

-useful for object detection, structure from motion, tracking

-Keypoint extraction to find interest points in an image (interest points / features

-keypoint descriptor describes detected interest points to make it feasible to match those across different images.

-keypoints themselves are a tool and used for many applications including object recognition.

-Feature detector examples:

-Harris Corner detector

-translation invariant (changing locations) -> yes

-rotation invariant -> yes

-scale invariant -> no

-Speeded up robust features – SURF

-Features from accelerated segment test - FAST

-Binary Robust Independent Elementary Features – BRIEF

-Oriented FAST and Rotated BRIEF – ORB

-Scale invariant Feature Transform – SIFT

-independently select interest points in an image with repeatable results

-extracts features that are most stable in terms of location and scale in an image, finding the best local maxima in regards to position and scale

-scale invariant -> yes

-Steps:

1. perform scale-space extrema detection, which is the process of

**-Past and current methods for scene stitching:**

Notes:

-to align images for image stitching;

1. find feature points in both images independently

2. find corresponding pairs between the two images

3. after finding the pairs, conjoin the images together using the corresponding pairs

# Methods and Algorithms:

Note: for the below include a snip of the code(?)

- Method for feature detection (and include keypoint detection) and why?

- Method for Matching and why?

- Method for box detection and why?

-Method for Image Stitching and why?

# Task 1 Table:

A yellow and pink background with black text

Description automatically generated

-Make a table of results showing number of true positive, false positive, true negatives, and false negatives for each scene.

-Calculate and show precision, recall, f1-score, and accuracy for the complete dataset. (Explain what conclusion can be made from the results).

-Note:

-when analyzing each scene, if an object was in the scene, but incorrectly or wrongly distributed with keypoints, then it is a false positive.

-if most ~70% of the object is not in the scene, then we consider it not in scene.

# Challenges Faced and Addressed:

-List all challenges that we faced and how we addressed them:

-optimization and how we changed the code (such as computing object keypoints once and sending them in an array to the function rather than computing on every iteration).

-issues with too many or too little keypoints (threshold affecting the keypoints focusing on detail that we faced).

-Challenges trying to get the boxes to properly go around the objects, can state that we did not discover a fix for this.

-Issues with grayscale output after SIFT operations and matching and had to make sure before each write that it was set back to color.

-Trying to ensure that in optimizing the code that we were pulling the correct objects and scenes in a good order -> dictionary was made to simplify this process

-The dictionary was also helpful for detecting boxes to ensure that the object real life name could be put onto the box to identify the object without looking up the id in the object folder.

# Possible Improvements and Future Work:

-likely not the best method using sift for this type of operation with the emergence of ML models that may perform better at faster speeds(?).

-would make more sense to use AI/ML models to identify these objects because the models can be trained on multiple objects intuitively rather than calculating and storing all keypoints of an object and assuming that they will be good for object identification and matching.

-I noticed on many of the matching scenes, either for true positives or true negatives, there are times where there are keypoints very far away from the object or sparsely placed (for the negative results)

-having a method to see the density of a an area and omitted completely sparsed keypoints would reduce the amount of true negatives and make the true positives produce better results.

# Works Cited

\*ensure that we site the opencv documentation that we used

-summarize image main:

[OpenCV: Feature Matching](https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html)

-boxes  
[OpenCV: Feature Matching + Homography to find Objects](https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html)

[OpenCV: Drawing Functions](https://docs.opencv.org/4.x/d6/d6e/group__imgproc__draw.html#ga5126f47f883d730f633d74f07456c576)