Takh2100 - Week 3 Task

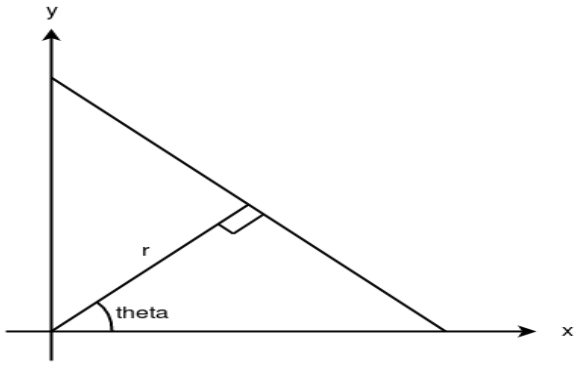
[code]

Introduction

This weeks task focused on Hough Space and Hough transformation, What is the benefits of the Hough space compared to spatial domain and how can we leverage this information to find the best line. The other part of the task was to implement a feature matcher where we match points in two images. We also Implement RANSAC for outlier detection.

Hough Transform

The Hough Transform is a powerful technique for detecting lines and other shapes in images, It operates by transforming the image from the spatial domain to a parameter space know as the Hough space. In the case of line detection, each point in the image is mapped to a curve in the Hough space, according to the equation $r = x\cos(\theta) + y\sin(\theta)$, where r is the perpendicular distance from the origin to a line, and θ is the angle between the x-axis and the line. This curve represents all the possible lines that could pass through that point.



When multiple points in the image lie on the same line their corresponding Hough curves should intersect at a single points. The value of the parameters on the intersection is the corresponding line where all the points lie.
The Hough transform is good because of its performance under noisy conditions where noise can make it hard to extract useful information in the spatial domain but in the hough space it is easier. It is also possible to represent all shapes that have a mathematical representation even vertical lines which is not possible in the coordinate domain.

Feature Matching and RANSAC

Feature matching is a technique used in computer vision used to identify corresponding points between two images, It relies on detecting key points such as corers or edges and computing descriptors to describe them. Algorithms like ORB, SIFT and SURF extract these

features. A matcher such as Brute Force which I chose to use pairs keypoints based on their similarity. To improve accuracy we use RANSAC to filter out incorrect matches by estimating transformations between images.

```
def apply_ransac(kp1, kp2, matches,
transformation_type=cv2.RANSAC, reproj_thresh=5.0):
    # RANSAC(Random Sample Consensus) is used to filter out
incorrect matches by
    # identifying the best transformation model that fits most of
the data while ignoring outliers.
    # If we don't have at least 4 matches, we can't compute a
valid homography.
    if len(matches) < 4:
        print("Not enough matches to apply RANSAC.")
        return [], None

    # Extract the coordinates of the matched keypoints from both
images.
    pts1 = np.float32([kp1[m.queryIdx].pt for m in
matches]).reshape(-1, 1, 2)
    pts2 = np.float32([kp2[m.trainIdx].pt for m in
matches]).reshape(-1, 1, 2)

    # Compute the homography matrix using RANSAC. This will
estimate the best transformation
    # while discarding incorrect matches (outliers) that do not
fit the model.
    H, mask = cv2.findHomography(pts1, pts2,
method=transformation_type, ransacReprojThreshold=reproj_thresh)

    # If RANSAC fails to find a valid transformation, return empty
results.
    if H is None:
        print("Homography could not be computed.")
        return [], None
    # The mask returned by findHomography marks inliers (correct
matches) with 1 and outliers with 0.
    # We filter out the inliers (good matches) based on this mask.
    inliers = [m for i, m in enumerate(matches) if mask[i]]

    return inliers, H
```