

ENPM673 Perception for Autonomous Robots (Spring 2024)

Project 3

Rohan Maan, Jay Prajapati, Samer Charifa, Tommy Chang

Submission guidelines:

- This homework is to be done and submitted individually.
 - Solve both the problems in **one Google Colab file**.
 - Your submission on ELMS/Canvas must be two files:
 - a. Google Colab (.ipynb) file
 - b. Google Colab (.pdf) file (Convert the same .ipynb file to a .pdf file following the naming convention ***YourDirectoryID_project3***).
 - If your email ID is **abc@umd.edu** or **abc@terpmail.umd.edu**, then your Directory ID is **abc**.
 - Ensure the code is well-formatted for readability.
 - Comment on each section (preferably every 2 lines) of the code to explain its purpose and functionality.
 - You can use any appropriate inbuilt OpenCV functions.
 - Provide clear and concise explanations for each step of the calibration process.
 - Include code snippets, plots, and visualizations to support your explanations.
 - Do not write the code in a single code cell.
 - Print the necessary output and images for the steps, while dividing the code as per the pipeline steps.
 - **You will lose marks if you don't follow any of the above guidelines.**
-

Problem 1: [50]

For this assignment, you are required to perform single camera calibration using either a chessboard or circular pattern. You will need to follow the steps outlined below and provide detailed explanations for each step of the calibration process.

1. Image Collection:
 - a. Collect approximately 50 images of a calibration board using a camera of your choice.
 - b. Save these images to your Google Drive. Also, add the link to these images in a text cell of your .ipynb file.
[Note: You might use the similar calibration board, but every individual should take their own 50 Images]
2. Calibration Pipeline:
 - a. Choose either a chessboard or circular pattern for calibration.
 - b. Explain the process of corner detection or centroid detection for the chosen pattern.
 - c. You do not need to write any function from scratch, Utilize OpenCV functions for camera calibration.
 - d. Provide a step-by-step explanation of each stage of the calibration pipeline and write code for it, including the initialization of calibration parameters, image undistortion, and optimization of camera intrinsic and extrinsic parameters.
[Note: Make sure you display the image and the undistorted image results.]
3. Reprojection Error Analysis:
 - a. Plot a graph showing the reprojection error for each image used in the calibration process.
 - b. The x-axis should represent the image number, and the y-axis should represent the reprojection error.
 - c. Discuss the significance of the reprojection error and its implications for the accuracy of the calibration.
4. Visualization of Calibration Results:
 - a. After calibration, redraw the detected corners or centroids on the original images.
 - b. Show the detected corners/centroids before and after calibration.
 - c. Differentiate between the original corner/centroid detection and the reprojection of these points based on the calculated camera parameters (R, T, and K).
 - d. Use different colors to represent the original and reprojected points for clarity.

Problem 2: [50]

Link to the images: <https://drive.google.com/drive/folders/1wkjEqhttgPs0nDvFVnsBeinE7ukYL2-f?usp=sharing>

You are provided with three datasets^[1], each containing two images of the same scene captured from different camera perspectives. Each dataset includes a calib.txt file detailing camera matrices, baseline, image size, and other parameters necessary for calibration and stereo vision.

Pipeline for Creating a Stereo Vision System:

1. Calibration:
 - a. Identify matching features between the two images in each dataset using any feature matching algorithms.
 - b. Estimate the Fundamental matrix using RANSAC method based on the matched features.
 - c. Compute the Essential matrix from the Fundamental matrix considering calibration parameters.
 - d. Decompose the Essential matrix into rotation and translation matrices.
2. Rectification:
 - a. Apply perspective transformation to rectify images and ensure horizontal epipolar lines.
 - b. Print the homography matrices (H1 and H2) for rectification.
 - c. Visualize epipolar lines and feature points on both rectified images.
3. Compute Depth Image:
 - a. Calculate the disparity map representing the pixel-wise differences between the two images.
 - b. Rescale the disparity map and save it as grayscale and color images using heat map conversion.
 - c. Utilize the disparity information to compute depth values for each pixel.
 - d. Generate a depth image representing the spatial dimensions of the scene.
 - e. Save the depth image as grayscale and color using heat map conversion for visualization.

Parameters in calib.txt:

cam0, cam1	Camera matrices for the rectified views, in the form $[f \ 0 \ cx; 0 \ f \ cy; 0 \ 0 \ 1]$
f	focal length in pixels
cx, cy	principal point
doffs	x-difference of principal points, doffs = $cx_1 - cx_0$ (here always == 0)
baseline	camera baseline in mm
width, height	image size
ndisp	a conservative bound on the number of disparity levels; the stereo algorithm MAY utilize this bound and search from $d = 0 \dots ndisp-1$
vmin, vmax	a tight bound on minimum and maximum disparities, used for color visualization; the stereo algorithm MAY NOT utilize this information

References:

[1] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In German Conference on Pattern Recognition (GCPR 2014), Münster, Germany, September 2014.