COS 529 Assignment 3: Bundle Adjustment

Due 11:59pm 11/21/2022 ET

Collaboration policy This assignment must be completed individually. No collaboration is allowed.

1 Bundle Adjustment

Bundle Adjustment the the joint non-linear refinement of camera and point parameters. Given a set of measured image feature locations and correspondences, the goal of bundle adjustment is to find 3D point positions and camera parameters that minimize the reprojection error.

In this homework assignment, you will be implementing bundle adjustment in python. You will be given a pickle file containing the following fields. The *Fixed* fields define the optimization problem and should not be modified. The *Variable* fields contain the variables which you will be optimizing over:

Fixed

- "is_calibrated": boolean flag whether or not the cameras in this problem are calibrated. If "is_calibrated==True", then you can assume that you are given the correct focal_lengths. If "is_calibrated==False", then you must also optimize over focal_lengths in addition to the other variables.
- "observations": a list of detected feature locations. Each observation is in the following form: `(camera_id, point_id, x, y)`
  - camera_id: index of the camera capturing the 3d point
  - point_id: index of 3d point being projected
  - (x,y) image coordinates of the projected point

Variable

- "poses": a list of camera extrinsics. Each entry in poses is the 3x4 extrinsics matrix of the corresponding camera.
- "points": a list of length num_points. Each entry in points is a 3d vector storing the location of the point in world coordinates.

- "focal_lengths": a list of length num_cameras. Each entry stores the focal length for the camera

Your assignment is to set up, and minimize an optimization problem over squared reprojection error of each of the observations.

$$E(\mathbf{P}, \mathbf{X}, f) = \sum_i ||\pi(\mathbf{P}[c_i], \mathbf{X}[p_i], f[c_i]) - (x_i, y_i)||^2 \tag{1}$$

where $\mathbf{P}[c_i]$ is the pose matrix for camera-id c_i, $f[c_i]$ is the focal length of camera c_i, and $\mathbf{X}[p_i]$ is the 3D-coordinate of point-id p_i. Equation 1 can be minimized using the nonlinear least-squares solvers such as the Levenberg Marquardt algorithm.

2 Camera Model

For this assignment, scenes are captured with the simple pinhole camera model. The projection function $\pi(\mathbf{P}, \mathbf{X}, f)$ projects 3D point $\mathbf{X}$ onto camera $\mathbf{P}$ with focal length f.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} f \cdot X'/Z' \\ f \cdot Y'/Z' \end{pmatrix}, \qquad \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2}$$

where $\mathbf{P} = [\ \mathbf{R} \mid t\ ]$. $\mathbf{R}$ is a 3x3 rotation matrix and $t \in \mathbb{R}^3$.

3 Task

You will be implementing Bundle Adjustment in python for this homework. The input to your system will be a pickle file containing the fields described above. The pickle file contains the initial values of "poses", "points", and "focal_lengths". Your code will optimize over "poses" and "points" (and "focal_lengths" if the problem is uncalibrated) to minimize the squared reprojection error. We have provided starter code `bundle_adjustment.py` and evaluation code `eval_reconstruction.py`.

- `bundle_adjustment.py` should take a pickle file as input, and output a new pickle file with the fields, "poses", "points", and "focal_lengths" replaced with their respective optimized values. You need to fill in the function `solve_ba_problem`. Which should return a new dictionary with the updated variables. Example usage:

  ```
  python3 bundle_adjustment.py --problem cube.pickle
  ```

  This command will output the solution file `cube-solution.pickle`.

- `eval_reconstruction.py` will evaluate the quality of your reconstruction on mean reprojection error. `eval_reconstruction.py` will also check to ensure that your camera extrinsics contain valid rotation matrices. We will check using the following criteria:

$$|det(R) - 1| \leq 10^{-5}$$
$$\max_{ij} |I - R^T R|_{ij} \leq 10^{-5}$$

  Solutions which do not contain valid rotation matrices will be considered incorrect. Example usage:

```
python3 eval_reconstruction.py --solution cube-solution.pickle
```

4 Grading

We have provided 4 test problems:

- sphere.pickle (18%)
- cube.pickle (18%)
- spiral.pickle (18%)
- obelisk.pickle (18%)

The first 3 problems, (sphere, cube, spiral) are captured using calibrated cameras, meaning that the focal-lengths provided are correct as given. In this case, you only need to optimize over the "pose" and "points" variables. The final problem, obelisk, is uncalibrated, meaning that you will need to include the "focal-lengths" in your optimization problem.

Your code will be evaluated on these 4 problems (18% each) and 2 unseen problems (14% each). On test cases, your solution will be considered correct if the mean reprojection error is less than 0.1 and all your poses are valid. On the unseen problems, your poses must still be valid, but you will be graded according to the final error achieved by your implementation according to the following function:

$$grade = \begin{cases} 14 \; points & \text{if error} \leq 0.1 \\ 14 * (0.1/error) & \text{otherwise} \end{cases} \tag{3}$$

Time and memory limits: Our reference implementation takes 5s and uses 60Mb of memory. For each problem, your implementation has a time limit of 10 min on a single CPU core and a memory limit of 1GB (time/memory determined by the AWS EC2 A1 machine as reference). Implementations which

exceed these limits on this machine will not receive points. For larger problems (obelisk) this means that your code will need to effectively leverage the sparse structure of the optimization problem in order to meet these constraints. Unseen problems will not be any larger than the provided test problems.

Library restrictions: In your implementation, you may not import any additional library that is not in the Python Standard Library: https://docs.python.org/3/library/. You may use numpy which is already imported in `bundle_adjustment.py`.

What to submit: Please submit to Blackboard the `bundle_adjustment.py` file with `solve_ba_problem` completed.