CS9645b Assignment 1

Due Friday February 21th 2020 at 11:00pm

Introduction

This assignment is a study on the noise robustness of a closed-form solution to the problem of cross-calibration between stereoscopic vision systems and 3D remote eye and gaze trackers. In many applications, it is essential to discover where the gaze of a person locates in a 3D stereoscopic depth map, and while published algorithms are iterative, there is a closed-form solution which employs an efficient algorithm for solving the Perspective-n-Point (**PnP**) problem.

In essence, the cross-calibration process estimates the homogeneous transformation matrix (a 3D rotation and a 3D translation) that transforms objects from the frame of reference of the tracker into the frame of reference of the stereoscopic vision system (and vice-versa with the inverse of the homogeneous transformation matrix).

Data Sources

The eye tracker provides a 3D point $e^T=(e_x, e_y, e_z)$ that indicates the position of one of the eyes of the observer, and a 3D gaze vector $\vec{g}^T=(g_x, g_y, g_z)$ describing the direction of the observer's 3D line of gaze. These elements are expressed in the frame of reference of the eye tracker. The stereoscopic vision system, on the other hand, provides 3D points $P_i^T=(X_i, Y_i, Z_i)$ expressed in its own frame of reference.

The first column of the data file contains the 3D points P_i in the reference frame of the stereoscopic vision system while the second column of the file contains the gaze vectors $\vec{g}_i$ in the reference frame of the eye tracker. The eye position e is not required (see below).

The PnP Problem

Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in world coordinates and their corresponding 2D projections in the camera image. The camera pose consists of a 3D rotation and 3D translation of the camera with respect to the world. Many solutions are available for the general case when $n>3$ and are available in the open source domain (you will be using Efficient **PnP**, or **EPnP** from OpenCV). Consider

$$\omega_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

succinctly expressed as $\omega_i p_i = M_c M_t P_i$, where p_i is the image point (in the camera frame

of reference) of P_i , which in turn is the 3D point in the world frame of reference. M_c is the matrix of camera intrinsic parameters, and M_t is the matrix that transforms a 3D point in world coordinates into the 3D point in the frame of reference of the tracker. A **PnP** algorithm estimates the matrix M_t .

The Cross-Calibration Process

In our case, the stereoscopic vision system provides 3D points P_i in its own coordinate system. However, the eye tracker provides only an eye position e_i and a gaze vector $\vec{g}_i$ for each P_i fixated by the observer. We do not have 2D image points corresponding to the 3D stereoscopic points. An effective way of solving this problem is to perspective project the vectors $\vec{g}_i$ onto a virtual projective plane perpendicular to the line of sight of the eye tracker at a distance of 1 from the projection center of the tracker.

The assumptions you are allowed to pose are:

1. The eye position points e_i are equal among themselves: $e_1 = e_2 = \ldots = e_n$. This means that the observer's eye position does not change as the observer is asked to gaze at 3D points P_i .
2. Since the eye position e_i is stable over calibration measurements, we can assume that gaze vectors $\vec{g}_i$ originate from the optical center of the eye tracker (this is also required in order to obtain the correct matrix M_t), instead of the eye position e_i . The net effect of this is a $-e_i$ translation of the 2D image points on the virtual image plane.
3. Since the projection plane for the eye tracker is virtual, then the optical center $(c_x, c_y)^T$ can only be equal to $(0,0)^T$ and, because the virtual projection plane is at distance 1 from the optical center, then the focal length parameters $(f_x, f_y)^T$ can only be $(1,1)^T$, resulting in matrix $M_c = I$, the identity matrix.

Also note the following important observation: $\omega_i p_i = \vec{g}_i$.

The Problem

The first step into the noise study of the above cross-calibration approach is to correctly estimate the matrix M_t , given the provided data file and the **EpnP** code from OpenCV. Once you have performed this step correctly, you may begin studying the noise properties of the algorithm.

In order to determine how this algorithm reacts to noise, we need to repeatedly introduce random, zero-mean Gaussian noise to the gaze vector data (the set of $\vec{g}_i$ measurements), with increasing values of σ for the Gaussian noise distribution. Proceed with the following steps:

1. set k to one
2. For each measurement $\vec{g}_i$:

- compute its magnitude m , and add a different zero-mean Gaussian random quantity generated with Gaussian spread σ equal to k % of m to each of the components of $\vec{g}_i$ to obtain noisy $\hat{g}_i$.
- Project $\hat{g}_i$ onto the virtual projective plane in order to obtain noisy $\hat{p}_i$

3. Using **EpnP**, compute the noisy matrix $\hat{M}_t$ with noisy points $\hat{p}_i$ and noiseless 3D points P_i
4. Undo the perspective projections on points $\hat{p}_i$ and obtain the 3D points into their noisy equivalents as $\hat{P}_i = \hat{M}_t^{-1}\hat{g}_i$. Note that $\hat{M}_t$ is not a square matrix. In order to compute the inverse transformation it represents, you need to add a 4th row to the matrix. This row is simply $(0,0,0,1)$.
5. Given n points, plot the average squared norm difference as $\frac{1}{n}\sum_{i=1}^{n}\|P_i - \hat{P}_i\|_2^2$. This is known as re-projection error
6. $k = k + 1$
7. If $k < 20$, repeat from step 2

To facilitate your task, you are given an implementation of a Gaussian random number generator.

What to Hand In

1. The source code for the program. The program must be well structured and documented. The program should output the noiseless matrix M_t in an easily readable format (no scientific notation for floats).
2. The graph of squared norm differences, for all noise levels considered, in pdf format.
3. A report with a presentation page, containing the matrix M_t , the graph, and your analysis of the noise results. This report must be well written and be in pdf format.
4. Submit these files with OWL, on or before the deadline.