Joshua Zapusek, Josue Perez, Aziz Boudy

CMPEN/EE 454

November 9, 2020

Project 2: Camera Projection Project

Part 1: Project Summary:

In this project we had the task to perform forward and inverse camera projection and perform all sorts of calculations along the way. The process of forward camera projection is the action of taking a 3D image and perform all sorts of transformations so that it would produce a 2D image. Inversely for inverse camera projection it consists of taking that 2D image and perform transformations, (in this case triangulation) to reconstruct a 3D image form the 2D image. The 3D points are represented as joints (i.e., shoulders, knees, etc.).

Through the duration of this assignment, we expected to need to interpret the mocap dataset for joint data, parse camera parameters, use those parameters to project 3D data into pixels for two image coordinate systems (pinhole model), to reconstruct the 3D locations of points (joints) using triangulation, to compute the Euclidean distances for final analysis, and to compute the algorithm efficiency.

In addition, we were tasked with computing the epipolar lines of the different views in the pinhole model.

We expect to glean further MATLAB experience and a deeper, practical understanding of Module 2 topics, including stereo vision, perspective, and motion capturing.

Part 2: Procedural Approach:

Attached to this document is P2_FlowChart.drawio. This document contains the general control flow of out MATLAB procedures. Below is an outline of each function withing main.

Main script:

Parameters: N/A

Outputs: N/A

Summary: We kept all functions in CMPEN_454_Project2_Script.m. The flow goes like that of P2_FlowChart.drawio. The main purpose is to read input data from video, to project frame numbers, and to each subsequent procedure. This process loops over all frames in input data.

Project3DTo2D:

Parameters: cam, worldCoord3DPoints

Outputs: projected2DPoints

Summary: This function takes cam (vue2 or vue4) and world coordinates for a specific frame in dataset and a 4x12 matrix pertaining to world joint coordinates. K matrix and P matrix are used in this procedure, which can be found inside vue2 & 4.

In order to compute pixel location for each joint, we use equations from "Relevant Equations". Then, obtain x, y points for pixel location be using equations for each joint and return the twelve pairs as a single matrix.

Reconstruct3DTo2D:

Parameters: cam1, cam1PixelCoords, cam2, cam2PixelCoords

Outputs: recovered3DPoints

Summary: We use triangulation to reconstruct 3D points from 2D pixel locations. Here, cam1 & 2 are vue2 & 4, and the pixel coordinates are the respective outputs from project3DTo2D function above for each camera perspective.

Calculating the viewing rays (equation under "Relevant Equations") for each camera requires a sequence of steps. First, we compute the elements of the "T" matrix covered in course lecture 15. We multiply each rotation matrix element by each respective position element (these parameters are found in the vue data structure). After we have "T" matrices (one for each camera), we use equations to calculate the location of the camera and the vector pointing from camera perspective to world coordinates. This yields Pw, the world point.

As was covered in lecture, we need to compute triangulation under the consideration of potential noise. We used lecture equations to compute the "U" vectors given in equations to solve for variables a, b, c. These "U" vectors are as follows: u1 & u2 are the unit vectors of the vector pointers (2D => 3D) outlined above, u3 is the cross product of the former. Then, simply solve the system of linear equations for coefficients a, b, c which are utilized for solving the midpoints between the two rays in equations.

This result yields the final computation of world point for a single joint. We repeat the above process twelve times.

FindEpipolarLines:

Parameters: World Coordinates, cam1 (e.i Rmat, Pmat), cam1PixelCoords, cam2 (e.i Rmat, Pmat), cam2PixelCoords.

Outputs: The output should be both images from either camera displaying marks on the joints and different color lines going thru those points. Going from where the other camera would be placed and looking from to the end of the image border

Summary: While the team fully understands the concepts and can solve for those epipole's coordinates and the functions for the epipolar lines. We tried our best to compute and display on the frames, but we fail to do so.

MeasureError:

Parameters: original3DPoints, recovered3DPoints

Outputs: recoveredError, being the SSD of both the original and recovered 3D points of the joints.

Summary: Take original point locations and, by way of sum of squares, take the Euclidean distance between for each joint location in 3D space. This operation occurs twelve times, one for each joint. Then, the Euclidean distances are averaged and returned.

Design Decisions:

Parameter Choice and Utilization:

To calculate the 3D=>2D transformations for vue2 & vue4, we simply passed the former along with our original world points to projeect3DTo2D function. This function was obviously called twice, once for each respective camera's perspective. Each vue held essential fields for calculating a 2D projection (Kmat & Pmat). To calculate reconstruction, we need Rmat from the vue field to compute world points. The remainder of procedures used derivations of these input data, which are detailed carefully in function outlines above.

Section 2.2 Q&A:

External Parameters: Focal length, Pmat, & Rmat

Internal Parameters: Orientation & position

Kmat: Film plane to pixels matrix and perspective projection matrix. Both are 3x3 matrices from equation # V

Pmat: Pmat is just the rotation matri x times the camera basis axis coordinates in world coordinate system to align the origins of world and camera position.

Location: The position for the camera is just the position field in world coordinates (i.e., vue2 = (-4450.1, 5557.9, 1949.1)).

Method of Demonstration:

For each function, we decided to simply display output matrices for projection and triangulation algorithms.

Relevant Equations:

Pixel location | Film plane to pixels | Perspective projection | World to camera | World point

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$$

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Equivalent, compact way to write this, commonly used in vision

$$\underset{3x1}{\lambda P_u} = \underset{3x3}{K} \underset{3x4}{[\underset{3x3}{R} \mid \underset{3x1}{t}]} \underset{4x1}{\begin{pmatrix} P_W \\ 1 \end{pmatrix}}$$

unknown scale factor ($\lambda > 0$)

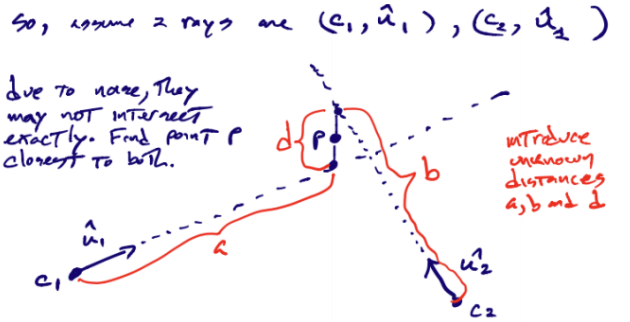
3x4 partitioned matrix

$$a P_l + c (P_l X R^T P_r) - b R^T P_r = T$$

$$O_l + a P_l \quad \text{and} \quad O_l + T + b R^T P_r$$

Robert Collins
CMPEN454

Specializing to our case



Point P must lie along line $\perp$ to both $\hat{u}_1$ and $\hat{u}_2$.

Let $\hat{u}_3 = \frac{\hat{u}_1 \times \hat{u}_2}{\|\hat{u}_1 \times \hat{u}_2\|}$

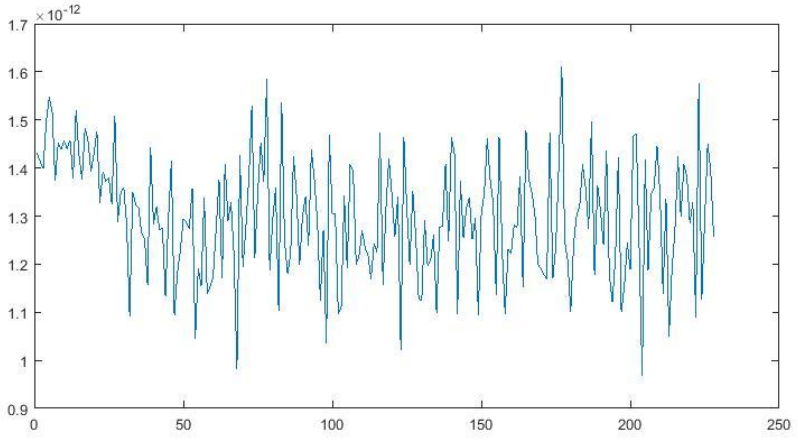
$a\hat{u}_1 + d\hat{u}_3 - b\hat{u}_2 = c_2 - c_1$ } 3 linear eqns in 3 unknowns a, b, d

After solving for a, b, d

$P_1 = c_1 + a\hat{u}_1 \quad P_2 = c_2 + b\hat{u}_2 \quad P = \frac{P_1 + P_2}{2}$

Part 3: Experimental Observations:

Below is an outline of outputs for each function detailed above.



(Plot of the frame id with the sums of square difference of the 3D plots from the original 3D image joints and the reconstructed 3D joins points)

Part 4: Quantitative Results:

The performance of the project was mainly evaluated by comparing and calculating the sum of squares difference between the original 3D points given before any forward and inverse camera projections against the 3D reconstruction after triangulation for our inverse camera projection.

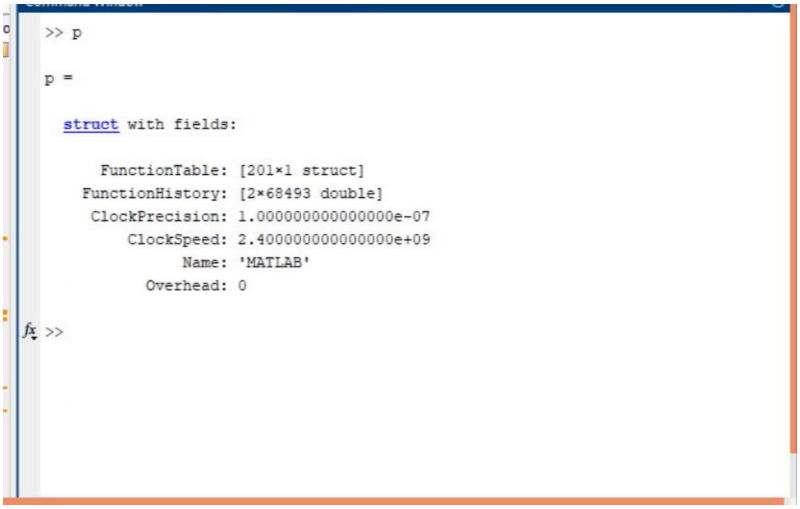
Part 5: Qualitative Results:

The performance of the project was mainly evaluated by comparing and calculating the sum of

Part 6: Algorithm Efficiency:

An image (below) indicates our algorithm efficiency. This reference was calculated using the MATLAB profiler output.

Below is the profile output for the script in aggregate. The clock speed was 2.4 MHz.



Part 7: Epipolar Visualization:

For our epipolar Visual

We couldn't do it. Too hard

Part 8: Team Members Contributions:

Joshua Zapusek: Report: wrote the introduction, procedure descriptions, flow-chart, and design decisions. Helped work through 3D projection algorithm with Josue and the data visuals

Josue Perez: Developed triangulation function and error measurements. Debugged the MATLAB script, answered 2.2 questions, and worked through displaying proper function outputs for given frame.

Aziz Boudy: No response from teammate