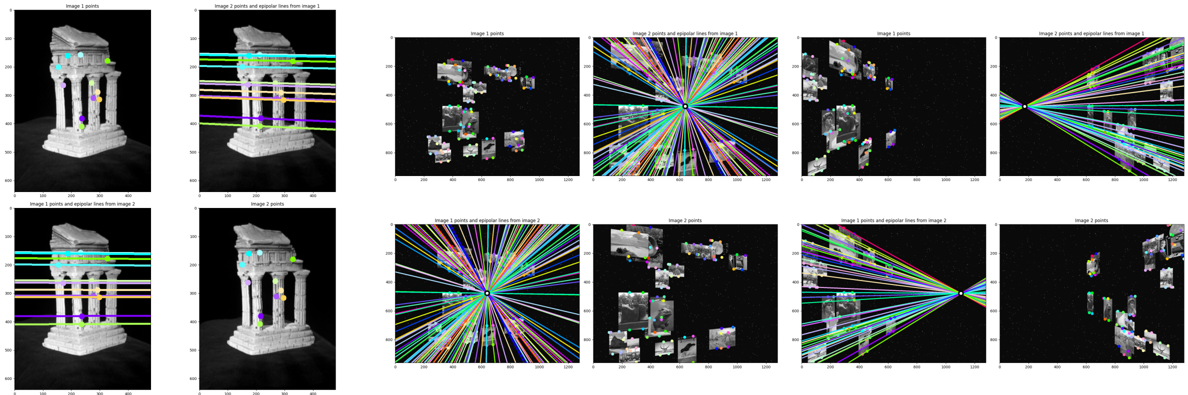
1 Camera Calibration



Temple zrtrans reallyInwards

Figure 1: Epipolar lines for some of the datasets.

Task 1: Estimating M [35 points]

We will give you a set of 3D points $\{\mathbf{X}_i\}_i$ and corresponding 2D points $\{\mathbf{p}_i\}_i$. The goal is to compute the projection matrix $\mathbf{M}$ that maps from world 3D coordinates to 2D image coordinates. Recall that

$$\mathbf{p} \equiv \mathbf{M}\mathbf{X}, \tag{1}$$

and (see foreword) by deriving an optimization problem. The script `task1.py` shows you how to load the data. The data we want you to use is in `task1/`, but we show you how to use data from Task 2 and 3 as well. **Credit:** The data from task 1 and an early version of the problem comes from James Hays's Georgia Tech CS 6476.

(a) (15 points) **Fill in `find_projection` in `task1.py`.**

(b) (5 points) **Report M** for the data in `task1/`.

(c) (10 points) **Fill in `compute_distance` in `task1.py`.** In this question, you need to compute the average distance in the image plane (i.e., pixel locations) between the homogeneous points $\mathbf{M}\mathbf{X}_i$ and 2D image coordinates $\mathbf{p}_i$, or

$$\frac{1}{N}\sum_{i}^{N} ||\text{proj}(\mathbf{M}\mathbf{X}_i) - \mathbf{p}_i||_2. \tag{2}$$

where $\text{proj}([x, y, w]) = [x/w, y/w]$. The distance quantifies how well the projection maps the points $\mathbf{X}_i$ to $\mathbf{p}_i$. You should use `find_projection` from part a). Note: You should feel good about the distance if it is **less than 0.01** for the given sample data. If you plug in different data, this threshold will of course vary.

(d) (5 points) **Describe what relationship, if any, there is between Equation 2 as above and Equation 6 in the HW5 Notes** Note that the points we've given you are well-described by a linear projection – there's no noise in the measurements – but in practice, there will be an error that has to minimize. Both equations represent objectives that could be used. If they are the same, show it; if they are not the same, report which one makes more sense to minimize. Things to consider include whether the equations directly represent anything meaningful.

2 Estimation of the Fundamental Matrix and Reconstruction

Data: we give you a series of datasets that are nicely bundled in the folder `task23/`. Each dataset contains two images `img1.png` and `img2.png` and a numpy file `data.npz` containing a whole bunch of variables. The script `task23.py` shows how to load the data.

Credit: `temple` comes from Middlebury's Multiview Stereo dataset. The images shown in the synthetic images are described in HW1's credits.

Task 2: Estimating F [35 points]

(a) (15 points) **Fill in** `find_fundamental_matrix` in `task23.py`. You should implement the eight-point algorithm. Remember to normalize the data and to reduce the rank of $\mathbf{F}$. For normalization, you can scale the image size and center the data at 0.

(b) (10 points) **Fill in** `compute_epipoles`**.** This should return the homogeneous coordinates of the epipoles – remember they can be infinitely far away!

(c) (5 points) **Show epipolar lines for** `temple`**,** `reallyInwards`**, and another dataset of your choice.**

(d) (5 points) **Report the epipoles for** `reallyInwards` **and** `xtrans`.