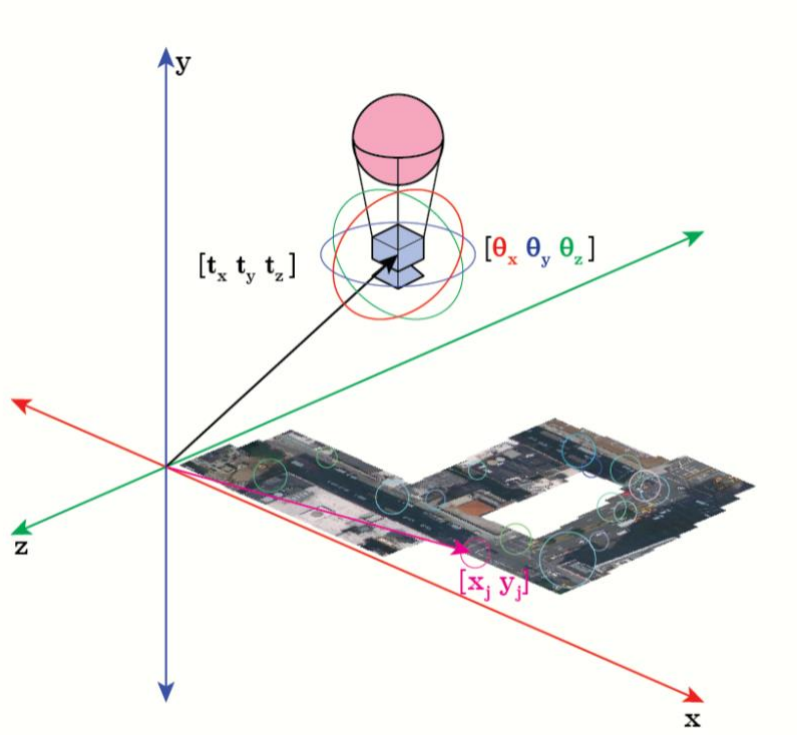
1. Identify the state space and how to parameterize the states. (10 points)

The state space includes the balloon-camera's extrinsics 3D-position and 3D-orientation relative to a world-frame origin as well as the camera's intrinsics. Each balloon state can be parameterized as:

$$\boldsymbol{b}_i = [t_x \;\; t_y \;\; t_z \;\; \theta_x \;\; \theta_y \;\; \theta_z \;\; f_x \;\; f_y \;\; c_x \;\; c_y]_i$$

The state space also includes the world-frame x- and y-coordinates of key features identified in all the images. Each image feature can be parameterized as:

$$\boldsymbol{k}_j = [x_j \;\; y_j]$$



2. Identify the measurement space. (10 points)

The measurement space is the set of observed features' locations in each camera image.

Let $\boldsymbol{z}_{ij}$ be balloon-camera's measurement/observation of key feature $\boldsymbol{k}_j$ when its state is $\boldsymbol{b}_i$

$$\boldsymbol{z}_{ij} = [x_{ij} \;\; y_{ij}]$$

3. Identify the prediction function. (10 points)

Given a camera's state (as described in problem 1) predict the world-frame locations of identified features:

$$k_j = (K_i\,[\,R_i \mid t_i\,])^{-1} z_{ij}$$

Where parameters from $\mathbf{K_i}$, $\mathbf{R_i}$ and $\mathbf{t_i}$ come from $\mathbf{b_i}$ as described in problem 1.

4. How do you find the initial states? And what is the optimization objective function (i.e. bundle adjustment) (10 points)

0. For the first image, establish world frame origin and initial position and orientation of the camera. Perform feature detection and generate vector descriptors.
1. For each subsequent image:
    a. For image taken at time, **t**, perform feature detection and generate vector descriptors.
    b. For the image taken at time, **t-1**, find common features with the same descriptors. For example:

time = t-1 time = t

    c. For each common feature, one can calculate the transformation, K [R | t], from image at **t-1** to image **t**, in camera space

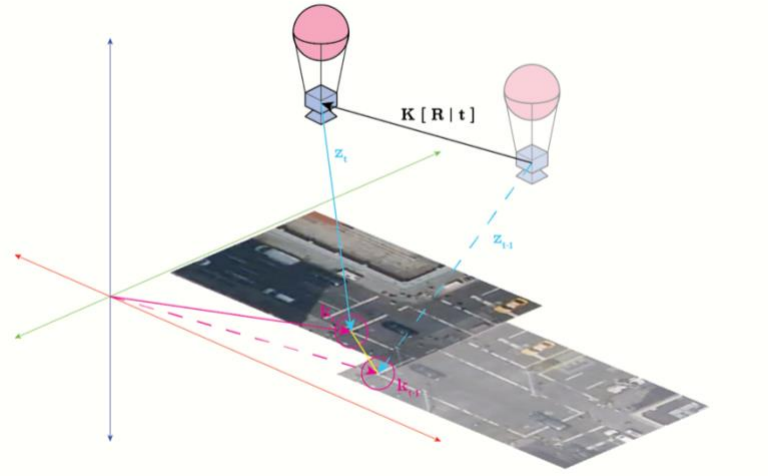
$$z_t = K[R \mid t]\, z_{t-1}$$

But since there is bound to be noise, solving the above equation will result in slightly different transformations for each feature.

$$z_j = K[R \mid t]\, z_{t-1} + \epsilon$$

One would have to find the "transformation of best fit" among all common features identified and minimize the error among the different measurements. In other words, the same transformation K [R | t]* will need to be applied to all previous common features and compared with the actual observations of those features in the current image:

   d. From the transformations of the common features in camera space, and accounting for the initial location of the camera and world frame origin, predict the 3D position and orientation of the camera and the common features' global locations at time t.
2. Each iteration of the above loop will have predictions of camera's global position and orientation and features' global locations.
3. These predictions have to be consistent. In other words, for any **two measurements** taken during two different times/images of the **same feature**, the error is the **distance between the global location of the feature predicted** by those two measurements. The optimization objective function is to minimize this error across all measurements from all images. In other words, adjust all **measurements** such that **error** between all feature's **world-frame location** estimates are minimized.



5. What methods/algorithms are suitable for automatically establishing the prediction function, and what are their advantages and disadvantages? (10 points)

In the case that the balloon is always parallel to the ground and on the same height, the descriptors will not be sensitive to scale. However, given that the images are of an outdoor environment, the descriptors will need to be invariant to illumination changes. Even though the balloon is at constant height, it can still rotate around the y-axis. Therefore, feature descriptors need to be rotationally invariant as well. Raw pixel values and filter bank descriptors do not meet these requirements.

The scale invariant feature transform (SIFT) algorithm generates illumination and rotation invariant descriptors, making it a better choice than raw pixel values and filter banks. As its name suggests, SIFT descriptors are also invariant to scale, which can come in handy if our balloon ever does vary in height. While it is a computationally expensive algorithm, this is not an issue for us given that high accuracy, rather than time-efficiency, is more critical for Sherlock. Should Sherlock ever grow impatient *when the game is afoot*, speeded up robust features (SURF) can achieve similar results in shorter times.

However, both SIFT and SURF are patented and cannot be used for commercial purposes. As CTO of YouDrive, I'd like to avoid getting into legal troubles, so open source feature detection solutions like BRIEF and ORB are preferred.

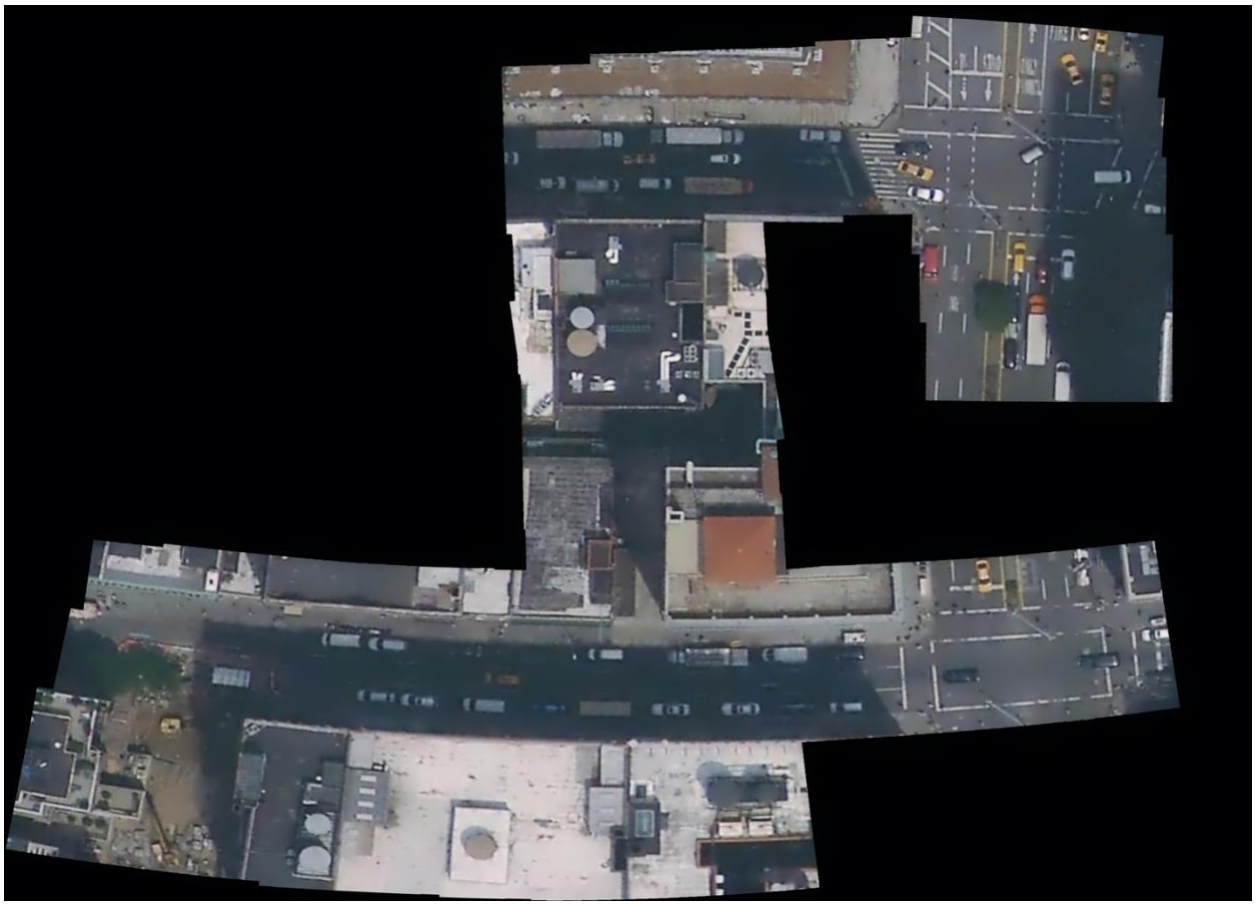
6. If the camera is not always parallel to the ground, which answer(s) to the above questions will change? And what would the change(s) be? (10 points)

Given that state space is parameterized by the camera's 3D rotation, none of the above answers should change except problem 4, where you will have to remove constraints rotation in calculating the initial state and optimization.

7. Write code to process the 41 images based on your answers above. Plot the initial states as well as the optimized states properly. (40 points)

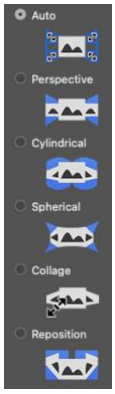
The `stitching_mod.py` file uses OpenCV's Stitcher class to form the map. However, it is unable to compose the map using all 41 images; after the first 37 images, Stitcher runs into camera parameter issues (see error code 3 in the Stitcher class documentation)

The following shows the result of using just the first 37 images (img_00000.jpg - img_00036.jpg):



Further adjustment of the Stitcher's bundle adjustment parameters may be necessary.

Without using code, I was able to stitch the images using Photoshop's photomerge functionality, which can either reposition the images or account for perspective distortions:



The following are results from Photoshop

Auto-merge Reposition Only

From the auto-merge result appears that Photoshop thinks some of the images are not taken completely parallel with the ground. Meanwhile, the reposition result shows problems with images 37-40, which corresponds to issues encountered with OpenCV.