Appendix- 2

Visual SFM Algorithm

Introduction

The approach used for the 3D reconstruction in the project is VisualSFM. VisualSFM is a GUI application for 3D reconstruction using structure from motion (SFM) by Changchang Wu. The reconstruction system integrates several of Wu's previous projects: SIFT on GPU(SiftGPU), Multicore Bundle Adjustment, and Towards Linear-time Incremental Structure from Motion. VisualSFM runs fast by exploiting multicore parallelism for feature detection, feature matching, and bundle adjustment.

For dense reconstruction, this program supports Yasutaka Furukawa's PMVS/CMVS tool chain, and can prepare data for Michal Jancosek's CMP-MVS. In addition, the output of VisualSFM is natively supported by Mathias Rothermel and Konrad Wenzel's SURE.

Algorithm

The approach used for the 3D reconstruction was to recover a set of camera parameters and a 3D location for each track. The recovered parameters should be consistent, in that the reprojection error is minimized (a non linear least squares problem that was solved using Levenberg Marquardt algorithm) Rather than estimate the parameters for all cameras and tracks at once, they took an incremental approach, adding one camera at a time.

The 1st step was to estimate the parameters for a single pair of images. The initial pair should have a large number of feature matches, but also a large baseline, so that the 3D locations of the observed points are well-conditioned.

Then, another image was selected that observes the largest number of tracks whose 3D locations have already been estimated. A new camera's extrinsic parameters are initialized

using the DLT (direct linear transform) technique inside a RANSAC procedure. DLT also gives an estimate of K, the intrinsic camera parameter matrix. Using the estimate from K and the focal length estimated from the EXIF tags of the image, a reasonable estimate for the focal length of the new camera can be computed.

The next step is to add the tracks observed by the new camera into the optimization. A track is added if it is observed by at least one other camera and if triangulating the track gives a well-conditioned estimate of its location. This procedure is repeated, one image at a time until no remaining image observes any the reconstructed 3D points. To minimize the objective function at each iteration, they used the Sparse Bundle Adjustment library. The run times for this process were a few hours (Great Wall - 120 photos) to two weeks (Notre Dame, 2635 images).

SfM using Two Images

Structure from Motion techniques using a pair of images were covered in the previous semester. The general technique for solving the structure from motion problem is to:

- estimate structure and motion up to a perspective transformation using the algebraic method or factorization method
    - estimate the m 2x4 projection matrices M_i (motion) and the n 3D positions P_j (structure) from the mxn 2D correspondences p_{ij} (in the ane case, only allow for translation and rotation between the cameras)
    - This gives 2 mxn equations in 8m+3n unknowns that can be solved using the algebraic method or the factorization method.
- convert from perspective to metric via self-calibration and apply bundle adjustment.

This project utilizes OpenCV:

- Compute fundamental matrix using RANSAC (OpenCV: findFundamentalMat())
- Compute essential matrix from fundamental matrix and K (HZ 9.12/9.13) OpenCV: Compute *E = K.T*F*K*
- Decompose E using SVD to get the second camera matrix P2 (HZ 9.19) (first camera matrix P1 is assumed at origin - no rotation or translation)

- Compute 3D points using triangulation

SfM using Multiple Images

With two images, we can reconstruct up to a scale factor. However, this scale factor will be different for each pair of images. How can we find a common scale so that multiple images can be combined?

One approach is to use the Iterative Closest Point (ICP) algorithm, where we triangulate more points and see how they t into our existing scene geometry. A second approach (and the one used on this project) is to use the Perspective N-Point (PnP) algorithm (also known as camera pose estimation) where we try to solve for the position of a new camera using the scene points we have already found. OpenCV provides the solvePnP() and solvePnPRansac() functions that implement this technique.

Multi View Stereo

The Multi View Stereo algorithms are used to generate a dense 3D reconstruction of the object or scene. The techniques are usually based on the measurement of a consistency function, a function to measure whether this 3D model is consistent with the input images"? Generally, the answer is not simple due to the e ects of noise and calibration errors.

Examples of consistency functions are:

- color: do the cameras see the same color? This approach is valid for Lambertian surfaces only and is based on a measurement of color variance.
- texture: is the texture around the points the same? This approach can handle glossy materials, but has problems with shiny objects. It is based on a based on a measurement of correlation between pixel patches.

One of the following two approaches are generally used to build the dense 3D model:

- build up the model from the good points. Requires many views otherwise holes appear.

- remove the bad points (start from the bounding volume and carve away inconsistent points). Requires texture information to get a good geometry.

There are usually several different 3D models that are consistent with an image sequence. Usually, we turn this into a regularization problem by assuming that objects are smooth. Then we optimize to nd the best smooth" object model that is consistent with the images.