CV 5369 SFM

Neel Bhalla

December 9 2021

Find the code and the `.ply` file at:
`ashdawngary/Tomasi-Kanade-Reconstruction`

1 Abstract

In this project, we explore a technique for reconstructing structures multiple images with significant overlap. Utilizing an orthorgraphic projection assumption(weak perspective model), It is possible to model the position of features along frames by the relative rotation of the true position of the feature wrt the centroid. In this project, we look into the Lucas-Kanade Pyramid OF algorithm, and the Tomasi Kanade factoring algorithm for image reconstruction.

2 Approach

The Approach is succinctly summarized by the Figure above. Given a set of frames, features are extracted from each of the frames. In order to correlate the features, a LK optical flow based detector takes the prior points, and provides estimations of corresponding features via the optical flow of the image.

Next an observation matrix is constructed where each row corresponds to one dimension of features for a given frame, and each column represents the values of the point for different dimensions and frames.

Finally given the matrix W, Tomasi Kanade Factorization is applied to extract the structure matrix, a $3 \times M$ stack of all the points that satisify W given the complement motion matrix (camera motion). Finally, the structure matrix is encoded into a point cloud with edges constructed to provide some constellations for the viewer.

KLT Good Features to Track

KLT Good Features to track a feature extraction algorithm that leverage the stability deciding metric in Lucas-Kanade method of image alignment. Given a image region for Lucas Kanade, The eigenvalues of such region determine whether it is homogeneous, an edge, or a

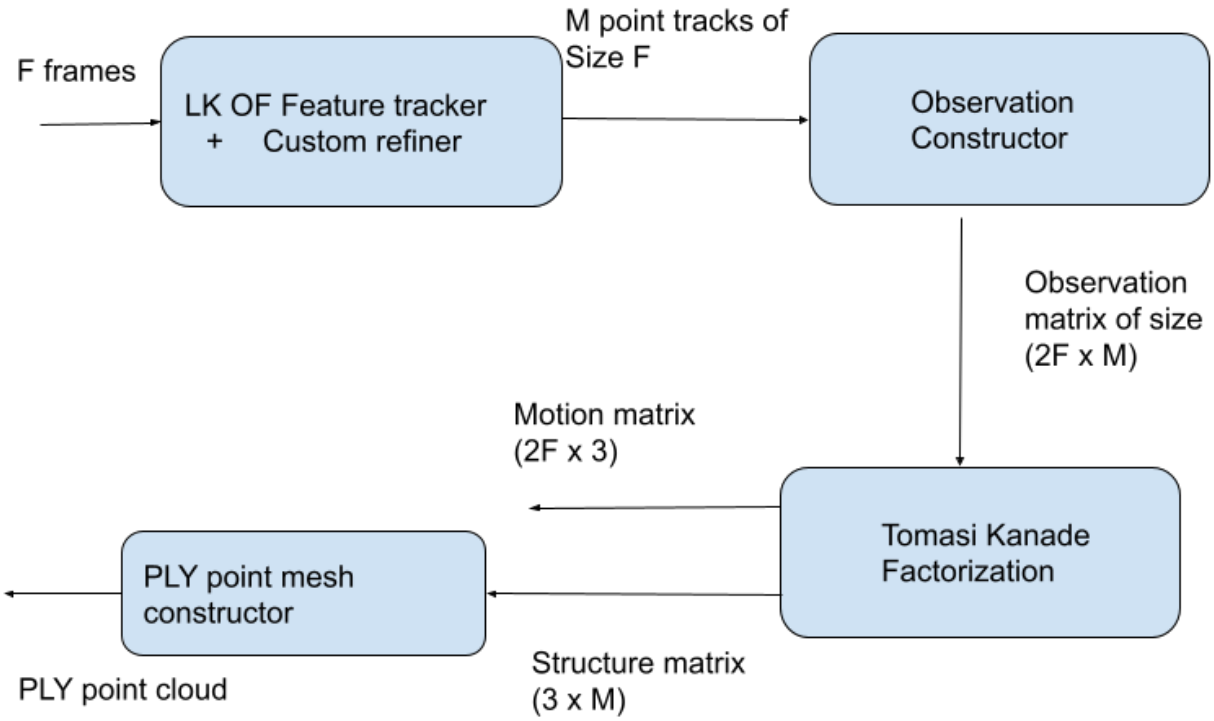

Figure 1: Pipeline of Frames

corner. KLT derives that in order to find a corner, we can determine a minimum threshold of the eigenvalues ($\lambda_{\min}$, and make sure that they are of the same magnitude to declare the area an actual corner.

LK OF Feature tracking

The Lucas Kanade Optical Flow algorithm that given two frames and a prior set of features, estimates the location of the feature in the new frame. LK is an iterative algorithm which starts by estimating a translation of 0. It iteratively calculates the error between the template feature and the current patch and terminates when the SSD is below a certain threshold. When the error is above, the patch is improved by adding the optical flow at such point to d, iterative-ly improving the estimate.

Pyramid updates

LK OF tends to fail for very large estimates, as it is effectively searching for a local minima for ssd-error. In order to decrease the search space, we can utilize a coarse-to-fine pyramid that allows estimates that might be large in fine levels to be manageable at smaller levels. The LK updates are corrected from fine scale to coarse scale.

2.1 Tomasi-Kanade Factorization

Recall the Affine Projection model:

$$x = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + b$$

Given that we normalize the coordinates of all our features (zero sum), we can elimate the translation component, meaning that $p = Ax$ is the model. This means that for every frame, we can represent the camera as an affine rotation that is centered at the centroid of the actual object of interest. Extending this to M frames yields:

$$W = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2F\,1} & x_{2F\,2} & \cdots & x_{2F\,n} \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_F \end{bmatrix} \begin{bmatrix} X_1 & X_2 & \dots & X_m \end{bmatrix}$$

where odd rows are x and even rows are y im cordinate measurements. It is worthy of noting W is of dimension $2F \times n$, A stack is of size $2F \times 3$, and the X stack is of dim: $3 \times m$.

Since we are able to factor W into two matrices, the rank of W is $\min(2F, 3, W)$ as the rank of the product of matrices is at most the min of the rank of the factors.

It suffices to perform an SVD of W, our observation matrix and keep the first three eigenvalues and eigenvectors, to enforce the rank 3 constraint. This gives us

$$W = MS$$

but observe for any invertible Q,

$$W = (MQ)(Q^{-1}S)$$

hence our factorization is not unique. Luckily, we have euclidean constraints on MQ, namely that the each row must be unit vector norm and pairs of vectors (x y bases) must be orthogonal:

$$(a_i^T Q)(a_i^T Q)^T = a_i^T Q Q^T a_i = 1$$
$$(a_j^T Q)(a_j^T Q)^T = a_j^T Q Q^T a_j = 1$$
$$(a_i^T Q)(a_j^T Q)^T = a_i^T Q Q^T a_j = 0$$

In order to solve it, let $L = QQ^T$. Observe that L must be *symmetric* as it is the product of a matrix and its transpose. This means that L is parametrizable by 6 entries as follows

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} L_{00} & L_{01} & L_{02} \\ L_{01} & L_{11} & L_{12} \\ L_{02} & L_{12} & L_{22} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = d$$

which can then be reduced to:

$$\begin{bmatrix} xx' & yx' + xy' & zx' + xz' & yy' & yz' + zy' & zz' \end{bmatrix} \begin{bmatrix} L_{00} \\ L_{01} \\ L_{02} \\ L_{11} \\ L_{12} \\ L_{22} \end{bmatrix} = d$$

Stacking all the constraints provides us a system that can be solved with least squares for the elements of L.

Once we have L it suffices to perform a cholesky decomposition to break it down as $L = QQ^T$ which gives us (MQ) and $(Q^{-1}S)$

3 Results

The file that constructs the `.ply` file is `p4_KLT_main.py` which produces `out_klt.ply` file. The vertices are specially colored to allow the user to figure out the orietnation (as factoring gives the structure up to an initial orientation ambiguity).

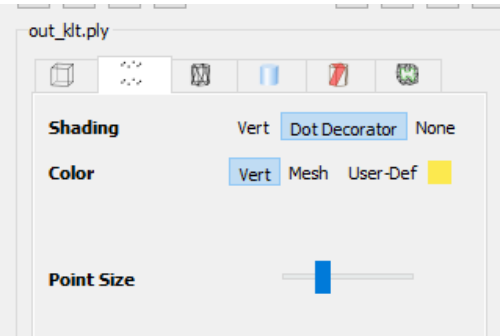

Figure 2: Make sure to set the vertex settings as such so the PLY colors are reflected with no shading effects.

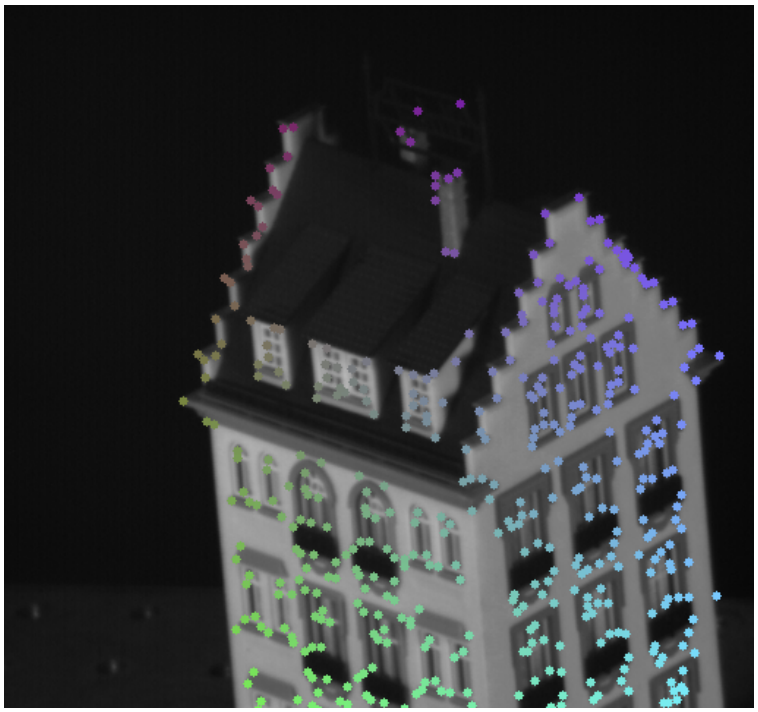

Figure 3: Last frame of the tracker (color coded for the PLY file)

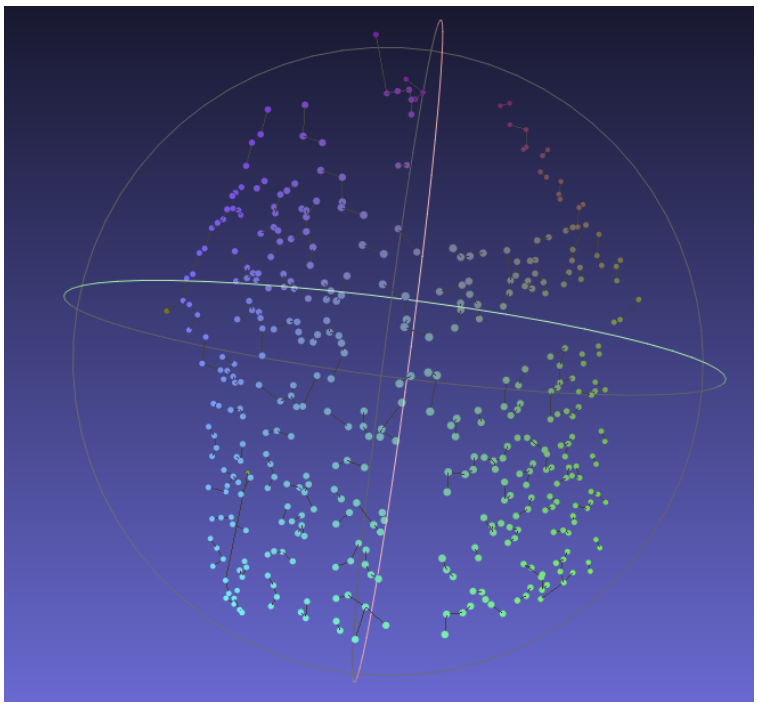

Figure 4: Isometric view of the hotel (orientation is flipped horizontally for some reason).

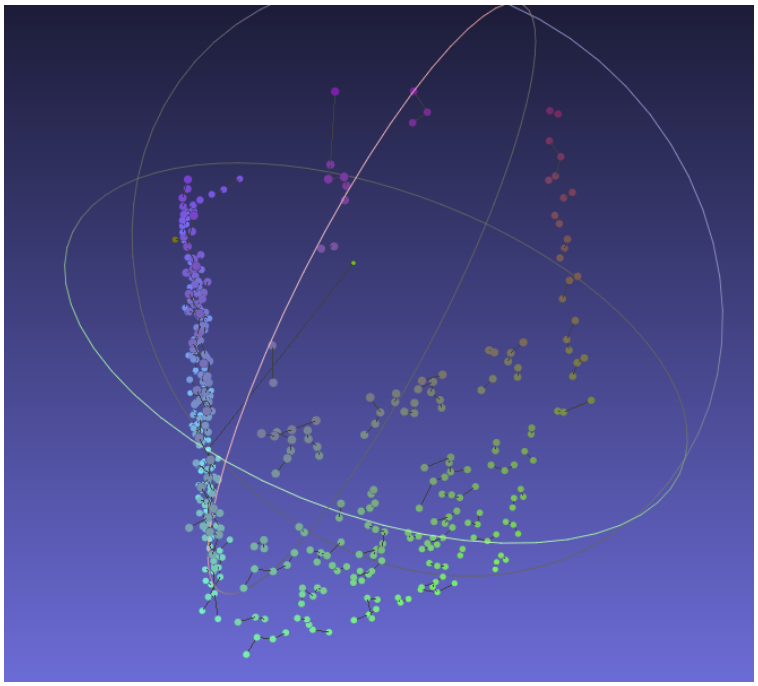

Figure 5: Top view of the hotel