Video Stabilization using Warping

Gupta, Ankesh
2015CS10435

November 28, 2017

Problem Statement

- Write the code for the recursive Gauss-Newton process yourself.
- Implemente video stabilization using above.

Implementation

1. For *image warping*, work of ***Lucas-Kanade 20 Years On: A Unifying Framework by Simon Baker and Iain Matthews*** was referred. **Inverse Compositional Image Alignment** was implemented.
2. For *video stabilization*, we first ask user for a bounding box/object that would be stabilised. Then all subsequent frames are *warped* to a reference frame(it was the *first frame* in our case).

Tips and Trick

Since the above was implemented in OpenCV using python, following are some tips and tricks for getting Gauss Newton to converge in the aforementioned language.

1. Use the *Inverse Compositional Image Alignment Algorithm* as mentioned above. Simple algorithm mentioned in section 2.2 of paper fails to converge.
2. Remove the *inverse* in the last step of iterative pseudo code.

3. Compute *Hessian Inverse* in pre-computation part for speedy iteration.

4. Avoid *for-loops* and compute all operations using in-built numpy functions for performance(Check reference code).

5. For video-stabilization, keep an iteration limit apart from ϵ, so that some frames that does warp(converge) gets handled.

Assumption

1. The stabilizer corrects motion upto a certain waver.

2. The object being stabilized does not move out of scene in subsequent frames.

3. Inverse of Hessian exists.

4. The transformation is affine.

From the previous section the basic outline of our method should be clear. Our first aim was to calculate the image of the absolute conic. Two approaches for this were considered. Firstly we examined the EXIF data to determine that the focal length used was 9 mm and we use the model number of camera to find that the sensor width was 5.76 mm and sensor height was 4.29 mm. Also the offsets of camera centre in image coordinate was found to be half the image width and height. In totality the camera matrix in pixel coordinates was found to be the following.

$$K = \begin{bmatrix} 3100 & 0 & 992 \\ 0 & 3121 & 744 \\ 0 & 0 & 1 \end{bmatrix}$$

Using this the image of the absolute conic and it's dual can be found with simple matrix operations as detailed in the theory part. The other approach was to use the orthogonality constraint equation shown earlier to solve for ω directly. This was abandoned after some tries since finding five orthogonal pairs of line from the image without any error was very hard. After this I found ω^* from the expression $KK^\top$.
Now we identified the three planes in the image and marked pairs of parallel

lines in the image in order to find the lins of infinity of the three planes. The marked image is shown in Figure 2. The three planes are - the desired slanting plane, the plane formed by the right side of the CSC building and the plane formed by the ground around the base of the staircase.
From the two points of infinity per plane the line at infinity of each plane was found out. The world coordinate system was assumed to be aligned to the CSC building (all three axes) with the sky in the +Z direction, the library in the +X direction and workshop in the +Y direction (only rough directions). Let α be the angle of the slant plane with the right side of the CSC building and β be the angle between the slant surface and the ground plane. Also let $\hat{n} =< a, b, c >$ be the desired normal vector. Then the following equations will give the desired results.

$$a = \hat{i} \cdot \hat{n} = cos(\alpha) = \frac{l_{slant}^{\top} \omega^* l_{csc}^{\top}}{\sqrt{l_{slant}^{\top} \omega^* l_{slant}^{\top}} \sqrt{l_{csc}^{\top} \omega^* l_{csc}^{\top}}}$$

$$c = \hat{k} \cdot \hat{n} = cos(\beta) = \frac{l_{slant}^{\top} \omega^* l_{ground}^{\top}}{\sqrt{l_{slant}^{\top} \omega^* l_{slant}^{\top}} \sqrt{l_{ground}^{\top} \omega^* l_{ground}^{\top}}}$$

$$b = \sqrt{1 - a^2 - c^2}$$

Results

I found the following values for $< a, b, c >$. The computations were done in MATLAB.

$$a = -0.0459$$

$$b = 0.219$$

$$c = 0.975$$

Thus the normal vector was found to be $< -0.0459, 0.219, 0.975 >$.

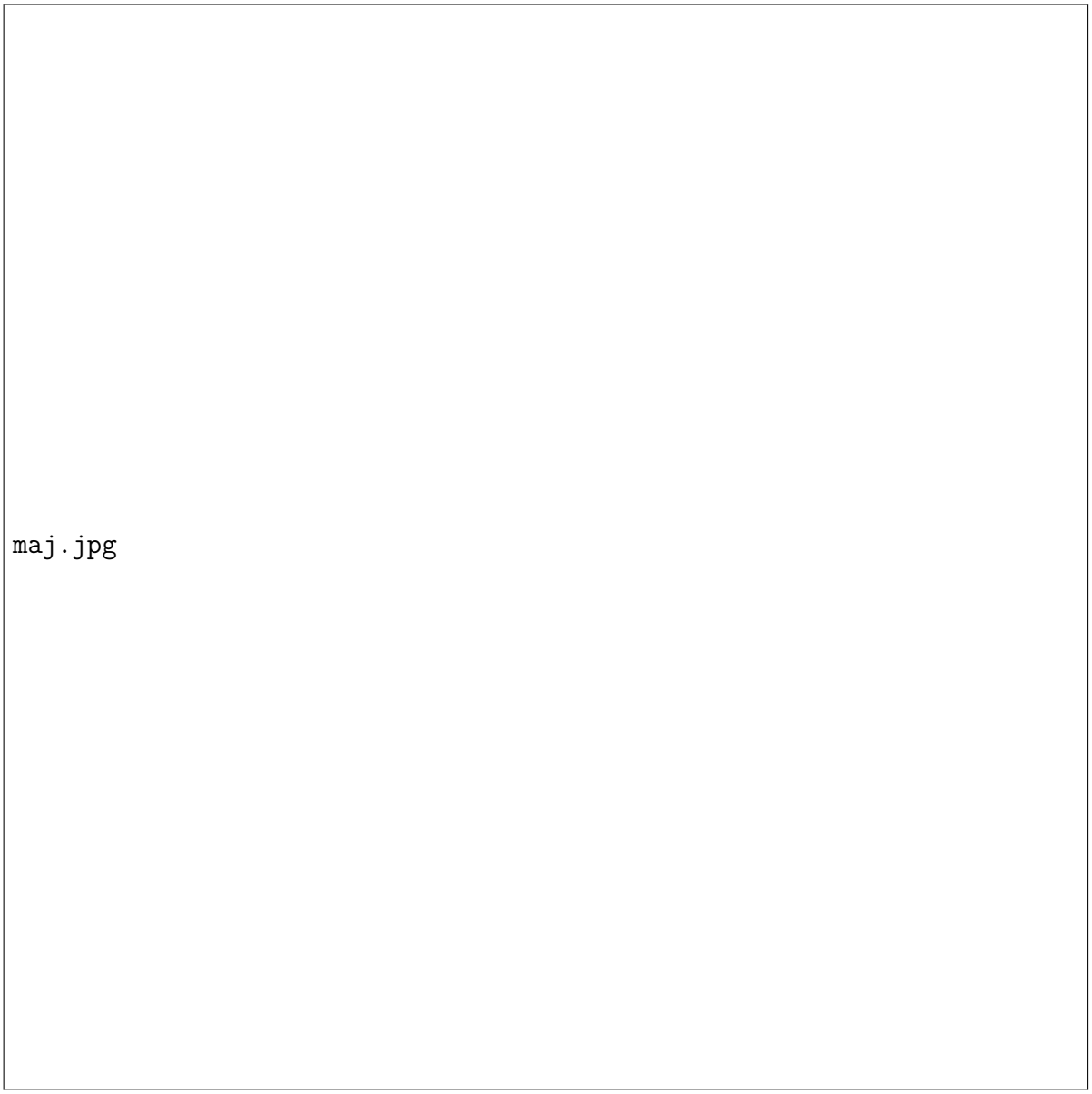

Figure 1: Marked parallel lines in three planes