Structure from Motion

Adebayo Eisape, Chris Simmons
Electrical and Computer Engineering
Johns Hopkins University
December 17, 2015

Structure from Motion

Adebayo Eisape, Chris Simmons
Electrical and Computer Engineering
Johns Hopkins University
December 17, 2015

Abstract

There are various methods that can be employed in reconstructing the three-dimensional geometry of a scene. Among these is stereo vision, which relies on the comparison of a real world scene from two camera point of views to reconstruct the scene in 3d space. The comparison gives the rotation and translation, which both constitute the overall transformation from one camera vantage point to the other. Through comparing the images, the disparity map, which gives the depth information from the images, is computed. Thus, with the pixel positions and depth, the 3D points are recovered and, ideally, a point cloud representation of the scene observed by the cameras could be made.

Project Process

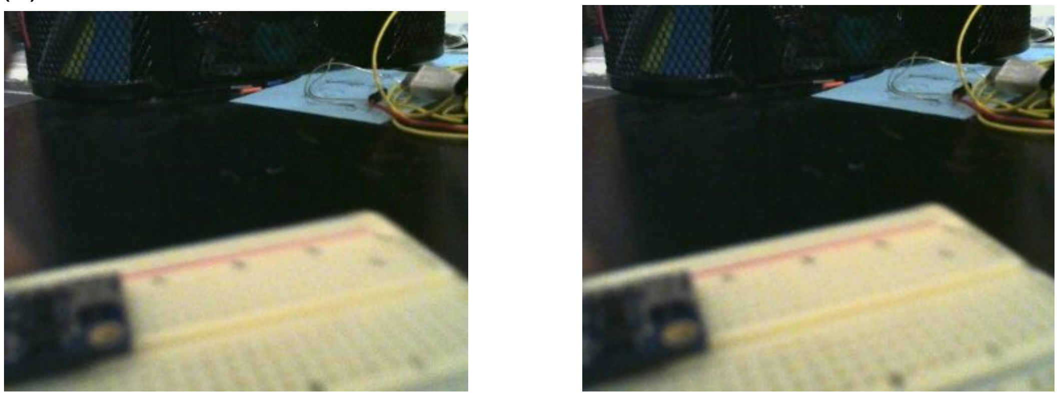
The first part of the experiment is to get the two images from the webcam, as displayed in Figure 1. From the images taken, the SIFT features are extracted in order to compose the fundamental matrix, which is responsible for corresponding matching points in the images. The fundamental matrix is also used to get the rectification transform of the images in order to project them on the same plane, which facilitates the disparity mapping. Thus, given the camera specifications, given in Figure 1, the calibration matrices for both images are computed. Since, in the project, the same camera is used to take both images and the assumption that both images are the same size, both the principal points and focal lengths are the same, which means the calibration matrices are the same. Having both the calibration and fundamental matrices the essential matrix computed, the essential matrix could be found. Through decomposing the essential matrix via SVD decomposition, the rotation and translation matrices are found.

Figure 1: Logitech c310 camera: focal length = 4mm, sensor size = 3.6 mm x 2.7 mm

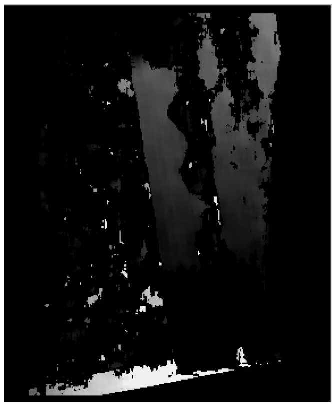
To get the 3D points from the images, two approaches were taken. The first approach was to use the rectified images to get disparity and use that to get the 3d coordinates for 3D reconstruction, as demonstrated below:

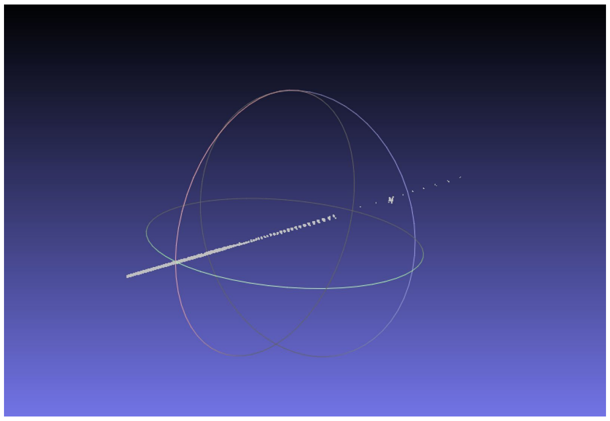
Figure 2: (a) The two input images taken from webcam, (b) depth image from both images (c) The 3D point cloud from disparity

(a)



Disparity


(b) Disparity image seem correct, could use some highlighting to illustrate the depth more.


(c) The 3D point cloud outputs a line of 2D planes, which is not an accurate reconstruction of the observed images.

The source of the incorrect 3d representation is uncertain; however, our matrices, with the exception of our essential matrix and, thus, rotation and translation, seem correct:

```
Fundamental Matrix:
[[ -1.46301755e-05   1.49251832e-02  -1.10748611e+00]
 [ -1.49013987e-02   3.45450795e-04   4.23724206e+00]
 [  1.11213781e+00  -4.28499344e+00   1.00000000e+00]]

Camera Matrix:
[[ 391.11111111    0.          176. ]
 [   0.         426.66666667  144. ]
 [   0.           0.            1.  ]]

Essential Matrix:
[[ -2.23794723e+00   2.49062612e+03   4.06429115e+02]
 [ -2.48665710e+03   6.28873981e+01   7.10118744e+02]
 [ -4.05284394e+02  -6.86257614e+02   2.25537821e+00]]

Rotation Matrix:                                           Translation Matrix:
[[ 0.8575105   0.0713497  -0.50949481]                        [[ 0.26605686]
```

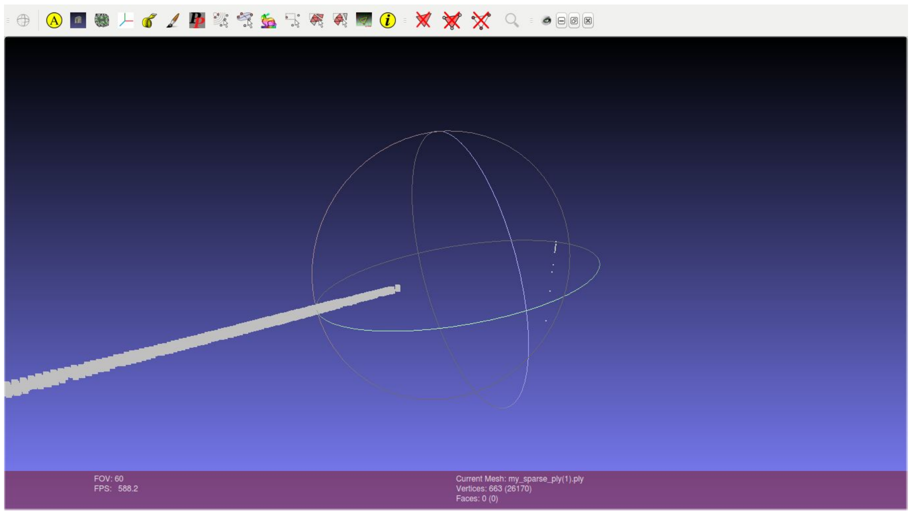
```
 [ 0.09421664  0.95180821  0.29186361]                        [-0.15529648]
 [-0.50576573  0.298279   -0.8094632 ]]                               [ 0.95136573]]
```

The inaccuracy of the essential matrix seems to be the where our project deviates from its ideal model. The second approach uses triangulation to get a more sparse representation of the 3D points. Unlike our first approach, the projection matrix is used and is constructed from the aforementioned rotation, translation, and calibration matrices:

```
Projection Matrix:
[[  2.46367116e+02   8.04027655e+01  -3.41734606e+02   2.71498162e+02]
 [ -3.26311638e+01   4.49057010e+02   7.96577138e+00   7.07368319e+01]
 [ -5.05765726e-01   2.98278997e-01  -8.09463199e-01   9.51365728e-01]]
```

Triangulation

Using the built-in opencv function to triangulate points, a ply file was created and displayed. this matches-only based approach seem to be more promising, as it yielded results different from any that were previously observed. Though the co-axial planes were still present, also present was a series of points that formed a line that intersected the axis of the coaxial planes (shown below). This gave reason to believe that this method was more reliable for our purpose of creating a 3d reconstruction from consecutive images.

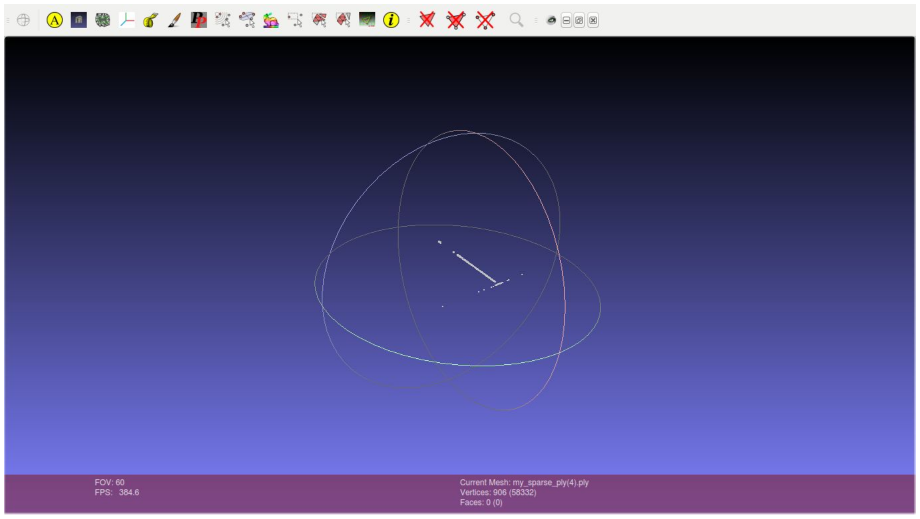


```
FOV: 60
FPS:   588.2
```

```
Current Mesh: my_sparse_ply(1).ply
Vertices: 663 (26170)
Faces: 0 (0)
```

d) Newly formed intersecting line

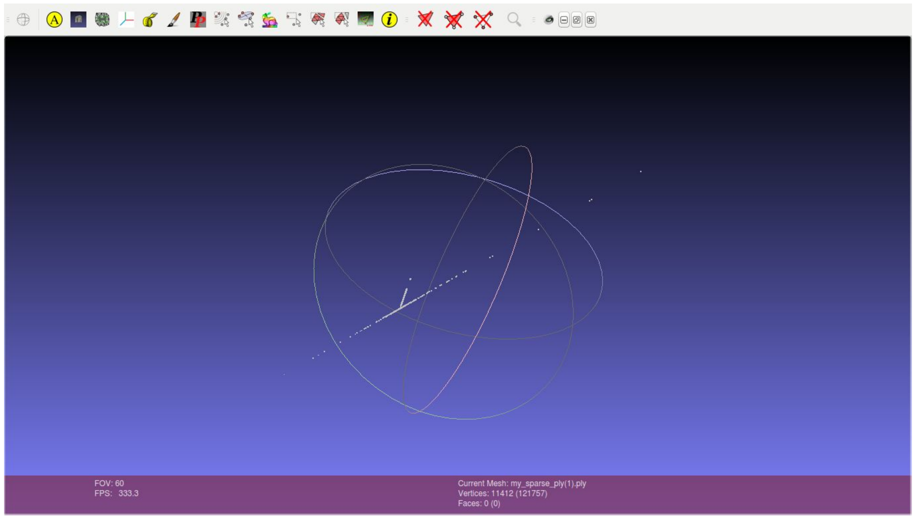
Our next step was to implement triangulation on a sequence of images, concatenating the resulting list of 3D points, though this proved difficult to effectively implement in practice. This was accomplished by computing a point cloud every two images and then warping the point cloud to the orientation of the reference cloud. For the reference cloud, the transform that encapsulates this change is the identity matrix. For later clouds, given the the cloud n and the reference cloud are not identical, the transformation expresses a rigid transformation (rotation and translation) between the two frames used to construct the point cloud. In order to combine the two point clouds, point cloud n must be transformed to the orientation of the reference point cloud. This is accomplished by multiplying the transforms between transform 0 (identity) and transform n. While our results improved slightly using several frames, the point cloud was still did not noticeably resemble any objects in the scene and only served to provide a denser representation of what was given by a single image.



FOV: 60
FPS: 384.6

Current Mesh: my_sparse_ply(4).ply
Vertices: 906 (58332)
Faces: 0 (0)

e) after five iterations



FOV: 60
FPS: 333.3

Current Mesh: my_sparse_ply(1).ply
Vertices: 11412 (121757)
Faces: 0 (0)

f) after 30 iterations

A phenomena we noticed with an increase in iterations was the shift in the line that intersect the main line of planes. This is most likely also due to our essential and thus rotation and translation matrices potentially being incorrect in addition to the drift of the transformation matrix when combining consecutive transforms.

We also noticed that the scene being observed did not significantly change the point cloud produced. Although the line of planes and intersecting line of points had different orientations in space and relative to each other, they did not produce the spread of three-dimensional points expected.

Conclusion

Though we we were not able to completely implement structure from motion or retrieve the position of the camera relative to the cloud (due to time constraint) we were able to produce a point cloud from consecutive frames of a video feed. improvements, however, can be made both in implementation and code used. The opencv function used to produce a disparity map does not produce a high quality one, and outliers are present in the matches used to compute the fundamental matrix.

Though structure from motion is appealing due to its seemingly simple implementation (one camera and some code), it is not ideal as it is affected by several factors that are not necessarily caused by the environment or easily changed (matches, floating point error, error propagation in the multiplication of consecutive transforms, etc), but have a significant impact on its performance if not in very specific settings with little variation.