Curtis Baker
cab365@cornell.edu
10/13/10

IPhone Based Fundamental Matrix Estimator

Introduction:

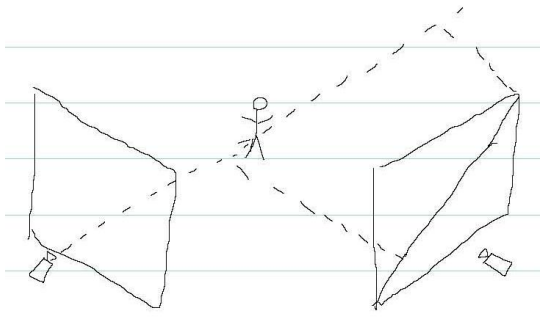
The fundamental matrix serves as the first step to a 3D stereo (or multiview) stereo reconstruction. It is a 3x3 matrix that maps points in one image to lines in another. At a simple level the fundamental matrix describes a relationship between 2 images that are viewing a common scene from different positions and angles. The goal of this project was to implement the functionality necessary to estimate the fundamental matrix of two images captured on the IPhone and to display some visual verification of the calculations.

Related Work:

The estimation of the fundamental matrix is not a new problem. There have been many implementations of it as well open source libraries (OpenCV) with functionality to compute these matrices. The OpenCV libraries that were used by this project are natively compiled for C/C++ and so some work needed to be done to make them usable in Objective-C. After several days of trying to get them to compile I found that Yoshimasa Niwa (http://niw.at/articles/2009/03/14/using-opencv-on-iphone/en) offers precompiled libraries for OpenCV. Though not directly implemented in this project a lot of the motivation behind this project was the research done by Georg Klein and David Murray. One paper of particular interest was 'Parallel Tracking and Mapping on a Camera Phone'.

The Heart of the F-Matrix:

The fundamental matrix related 2 images by their epipolar geometry. That is, when a point is multiplied by the matrix a line is returned. This line is the projection of ray:camera1->point1 onto the image plane of camera2.



The decomposition of the fundamental matrix gives: $F = K_2^{-T} R [t_x] K_1^{-1}$

K2 = Camera 2 intrinsics; R = Camera 2 rotation; t[x] = position of camera 2; K1 = camera 1 instrinics.

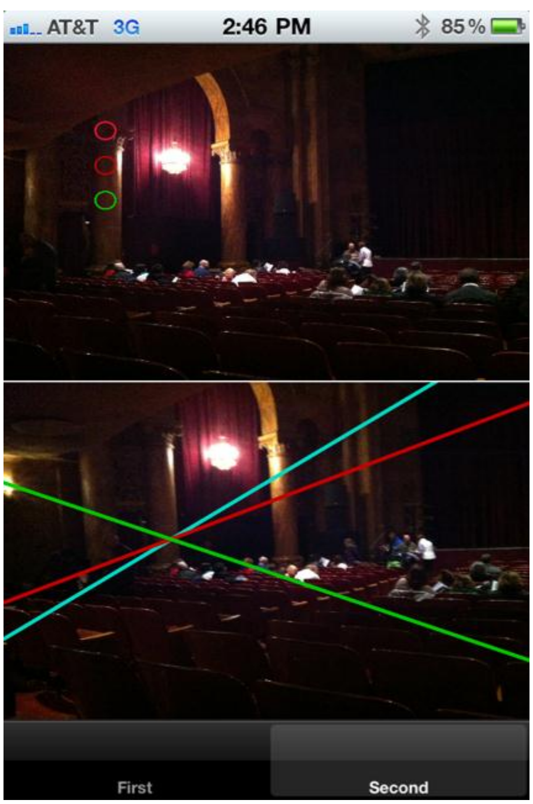
Corresponding points in the 2 images are constrained on: $x'^T F x = 0$

Approach:

1. Capture 2 images of the same scene with a translation and slight inward rotation.
2. Filter and scale images to promote faster computations on the A4 processor.
3. Compute the SURF features and descriptors for each image using the OpenCV function cvExtractSurf.
4. Match the features using a SSD based nearest neighbor function and store SSD distance.
5. Compute the score for each match based on ratio score proposed in class and used in project 2.
6. Compile matching point matrices.
7. Estimate the fundamental matrix using cvFindFundamentalMat.
8. For each match display the feature colored and sized based on the score of the match. The brighter and bigger represent not only a less ambiguous match (low ratio score) but a better match as well (smaller SSD distance).
9. Compute epipolar lines using cvGetEpipolarLines
10. Display image with features and epipolar lines generated.

Results:


BIG LEBOWSKI
HOT FUZZ
PINEAPPLE EXPRESS
MUNICH
snatch
THE KINGDOM


AT&T 3G
2:46 PM
85 %
First
Second

Discussion:

The results on my initial prototype (computer based) in c++ were great as can be seen in the first set of images. The application consistently found good fundamental matrix estimations and the corresponding epipolar lines were right on. The phone version however had several issues. As can been seen in the second image of it running on the phone itself, the matrix itself is in accurate and further investigation revealed an issue with the matching. There were known bugs within the cvExtractSurf function particularly in the version of OpenCV for which I could obtain compiled libraries for. One bug that caused several problems within this function was the lack of robust handling for low feature counts. If an over or under exposed image is presented to the function it will frequently throw an exception and abort. This was particularly frustrating because it did not even give you the option to handle the case where no features were detected.

Experimentation on the desktop version yielded results that showed the RANSAC method providing more accurate estimations but at a slower rate (depending on desired reliability). The 8-point method and LMedS method appeared to be in both speed and accuracy. Little experimentation was done on the 7-point method but it indicated a similar speed to the 8-point but would return 3 matrix estimations which would require additional computing to choose the best of the 3 presented.

Future Work:

In the future (and perhaps for my Masters project) I would like to implement an position tracking and augmented reality system similar to the works of Klein and Murray. An interesting extension to their work would be the integration of their position tracking into the camera controls of my IPad based physics simulation.