Exercise in Estimating Camera Motion and Structure

Henrik Aanæs, haa@imm.dtu.dk

April 8, 2013

This is the second half of exercise nine in 02504

In this exercise you should estimate the camera motion and a sparse 3D structure related to three images. These images, `Im1`, `Im2` and `Im3` together with a given internal calibration matrix, `K`, should be found in the MatLab file `CamMotionData`. Accompanying this exercise you should also find the mex function `EestMex.mexw64`. The exercise runs in the following steps, and you should illustrate the results as you go along.

1. Extract SIFT features from the three images using vl_feat, e.g. by using the following command
   `[F1, D1] = vl_sift(Im1,'FirstOctave',1);`

2. Make a function
   `[M12,E,R,t]=MatchImagePair(F1,D1,F2,D2,K,Sigma)`
   which from two sets SIFT feature and descriptors (F1,D1) and (F2,D2) the internal parameters K and a noise estimate Sigma , computes:

   - The matches, M12, between the two feature sets. M12 should be a two by the number of matches matrix, containing pairs of indices for matches.
   - An estimate of the essential matrix, E, the rotation, R, and the translation, t, between the two cameras.

   Do this in the following manner:

   (a) Compute initial matches via `vl_ubcmatch` from the vl_feat package.

   (b) Extract the matching 2D feature points for the two images, q1 and q2. That is that `q1(:,10)` should be the match for `q2(:,10)` etc. The q1 and q2 should be in normalized homogeneous coordinates, i.e. put a one beneath them.

   (c) Use the supplied mex function EestMex to estimate the essential matrix, rotation and translation between the cameras, by the following call
   `[E,R,t,nIn]=EestMex(K,q1',K,q2',Sigma);`
   This function does a robust estimation of the essential matrix via RANSAC, and decomposes the matrix into a rotation and translation.

   (d) Form the fundamental matrix
   `F=inv(K)'*E*inv(K);`
   and compute which matches are consistent with the found epipolar geometry using the Sampsons distance.

   (e) return the consistent matches in M12.

3. Match image one and two as well as image two and three via the following calls (Sigma should be 3)

```
[M12,E12,R12,t12]=MatchImagePair(F1,D1,F2,D2,K,Sigma)
[M23,E23,R23,t23]=MatchImagePair(F2,D2,F3,D3,K,Sigma)
```

4. Compute the 2D features that are matches over all three images, e.g. using the function call: `[ ,Idx12,Idx23] = intersect(M12(2,:),M23(1,:));`

5. Compute the corresponding 2D feature points q1,q2 and q3, such that `q1(:,10)`, `q2(:,10)` and `q3(:,10)` correspond to the same track or 3D feature, etc.

6. Form camera one and two, setting camera ones external parameters equal to the global coordinate system, i.e. R1=I and t1=0.

7. From the tracked features from image one and two (use only the points tracked through all three images) compute the 3D points, Q.

8. Write a function for camera resectioning. NB: normalize wrt. the 2D feature points.

9. Based on the computed 3D points and the tracks in the third image, q3, estimate the third camera, Cam3.

10. Based on Q, and Cam3 compute the projections into the third image, i.e. (remember to normalize)
    `p3=Cam3*Q`

11. Compute the reprojection error, i.e. the unhomogeneous distance between q3 and p3.

12. Recompute Cam3 using only the points with a reprojection error of less than 30.

13. Do one more iteration 9. to 12. This should give robustness towards outliers in the 3D points.

14. Ideally a RANSAC approach with the camera resectioning should have been use. Describe how this should be constructed.

15. If a fourth image/camera should be addded, how would you do that?