Introduction to Computer Vision Homework 4

Jeeeun Kim (jeeeun.kim@colorado.edu)

Due in two weeks in class. All renderings should be done in Matlab, saved as some form of scalar vector graphic (such as ps, eps, or pdf) and included in your final .pdf report (typeset in LATEX).

1. Get the data and code:
(a) Download and compile VLFeat
(b) Run vl_demo_sift_match – read the code.
(c) Download sequence of images from: https://arpg.colorado.edu/wpcontent/uploads/cityblock.zip
(d) Compile calibu's matlab camera model wrapper (run ccmake in calibu and enable matlab code)
(e) Add calibu's matlab directory to your matlab path; e.g., addpath(<path_to_calibu_mex_library>)

2. Write a matlab program StereoVisualOdometry.m that
(a) Use matlab command calibu_rig('cameras.xml') to load the camera rig extracted from cityblock.zip
(b) Loads a list of all the left and right images;
    e.g., something like limages = dir('left*'); rimages = dir('right*');
(c) Loop over all stereo pairs extracting SIFT features and matching them between left and right images; draw the correspondences

```
    rig = calibu_rig('/home/viki/Downloads/cityblock/cameras.xml');

    lfiles = dir('/home/viki/Downloads/cityblock/images/left*');
    rfiles = dir('/home/viki/Downloads/cityblock/images/right*');
    n = size(lfiles);

    xl = []; yl = [];
    xr = []; yr = [];

    for ii = 1:n
       limg = imread(strcat('/home/viki/Downloads/cityblock/images/',
lfiles(ii).name));
       rimg = imread(strcat('/home/viki/Downloads/cityblock/images/',
rfiles(ii).name));


       [fl, dl] = vl_sift(im2single(limg_prev)); % f(1:2) = the descriptor pos(x,y)
       [fr, dr] = vl_sift(im2single(rimg_prev)); % d = descriptors of corresponding
frame f

       %matches = index of descriptors (row size: # of descriptor, 128 at iteration 1)
       %scores = match score of this point
       [matches, scores] = vl_ubcmatch(dl, dr); % match between left and right frame
       [drop, perm] = sort(scores, 'descend');


       matches = matches(:, perm); %index of features
       scores = scores(perm);

       figure(1);
       imshow(cat(2, limg, rimg)), 'Border', 'tight');
```

```
        % get x,y positions of match descriptors
        hold on;
        xl = fl(1, matches(1,:));
        xr = fr(1, matches(2,:)) + size(limg_prev,2); % move right frame right
        yl = fl(2, matches(1,:));
        yr = fr(2, matches(2,:));

        % draw lines between left and right frame
        h = line([xl;xr], [yl;yr]);
        set(h, 'linewidth', 1, 'color', 'b');
        plot([x1;xr], [yl;yr], 'c');

        waitforbuttonpress;
        clf;
end
```

Figure 1. Code to read series of left and right frames, and match features between the two

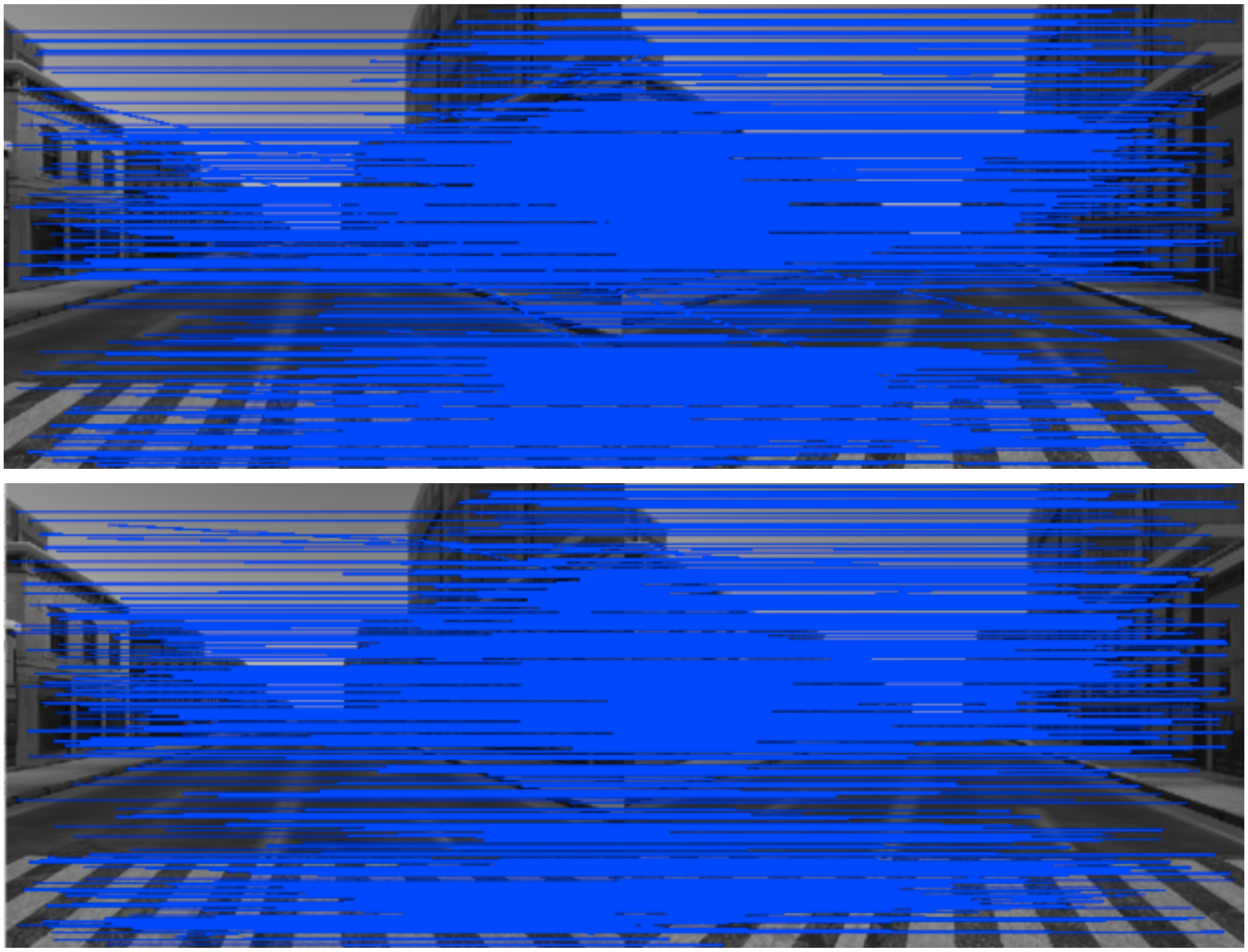


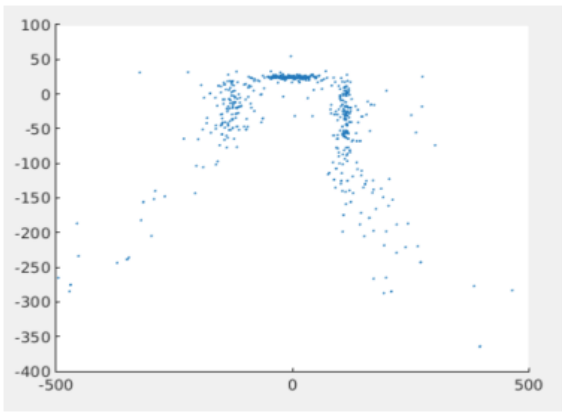
Figure 2. Matches between left and right frame at iteration 1 (top), and iteration 5 (bottom)

3. Triangulation

(a) Triangulate all correspondences; draw the 3D points and draw the camera poses.

```
for ii = 1:n

    uvl = fl(1:2,matches(1,:)); % compress xl, yl into uvl
    uvr = fr(1:2,matches(2,:));

    cmodl = struct('fx',268.5119, 'fy', 268.5119, 'cx', 320.5, 'cy', 240.5);
    cmodr = struct('fx',268.5119, 'fy', 268.5119, 'cx', 320.5, 'cy', 240.5);
    Twca = [ 1, 0, 0, 0.2; 0, 1, 0, 0.05; 0, 0, 1, -2.0 ];
    Twcb = [ 1, 0, 0, 0.2; 0, 1, 0, 0.75; 0, 0, 1, -2.0 ];

    figure (2); hold on;
    %index == inliers
    [p, index] = triangulate(uvl_prev,uvr_prev,Twca,Twcb,cmodl,cmodr)
    plot3(p(1,:), p(2,:), p(3,:), '.');

    waitforbuttonpress; clf;
end

function [p, index_keep] = triangulate(uvl,uvr,Twl,Twr,lcmod,rcmod)

    d = uvl(1,:) - uvr(1,:);
    index = find(d>10|d<0.1);        % outlier
    index_keep = find(d<=10|d>=0.1); % inliers, keep indexes for later RANSAC

    d(index) = [];

    uvl(:,index) = []; %clean outlier indexed elements
    uvr(:,index) = [];

    b = norm(Twl(1:3,4) - Twr(1:3,4));
    f = lcmod.fx;

    p(3,:) = b*f./d;
    p(1,:) = (uvl(1,:) - lcmod.cx).*p(3,:)./f;
    p(2,:) = (uvr(2,:) - lcmod.cy).*p(3,:)./f;
end
```

Figure 3. Code for triangulate 3D points from two pairs of uv points



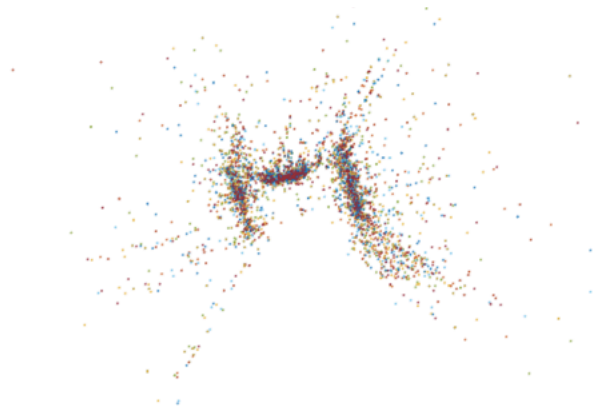


Figure 4. Triangulated 3D points from iteration 1 (left),
and accumulations from several iterations (right, blue:fr1, red:f2, orange:fr3 … green:fr10)

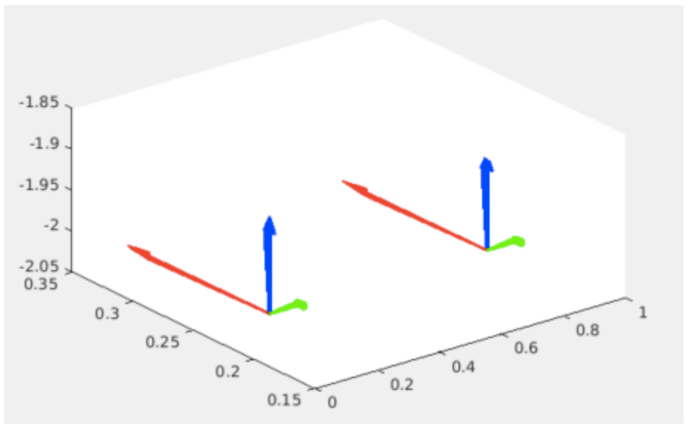


Figure 5. Two camera poses from left (lower left) and right (upper right)

4. Tracking

(a) Find correspondences between current and previous frame for all triangulated 3D points from the previous frame; Display a 2x2 grid of images, current images on the top row, previous frame on the bottom row, left images in the left column, right images in the right column. Draw all correspondences.

```
[rl,cl] = size(dl_cur);                   % clean current frame’s descriptor(top)
for ii = 1:rl
    for ij = 1:cl
       if dl_cur(ii,ij) ~= dr_cur(ii,ij)  % where does not match left and right
          dl_cur(ii,ij) = 0;
          dr_cur(ii,ij) = 0;
    end
end

% new match between left previous ~ current frame with cleaned dl_cur above
[matches_new_v1, scores_new_v1] = vl_ubcmatch(dl_cur, dl_prev);
[drop, perm_prev_new_v1] = sort(scores_new_v1, 'descend');
matches_new_v1 = matches_new_v1(:, perm_prev_new_v1);
scores_new_v1 = scores_new_v1(perm_prev_new_v1);

uvl_prev_new = fl_prev(1:2,matches_new_v1(1,:)); % find new pos x-,y- matched
uvl_cur_new = fl_cur(1:2,matches_new_v1(1,:));

% new match between left previous ~ current frame with cleaned dr_cur above
[matches_new_v2, scores_new_v2] = vl_ubcmatch(dr_cur, dr_prev);
[drop, perm_prev_new_v2] = sort(scores_new_v2, 'descend');
matches_new_v2 = matches_new_v2(:, perm_prev_new_v2);
scores_new_v2 = scores_new_v2(perm_prev_new_v2);

uvr_prev_new = fr_prev(1:2,matches_new_v1(1,:)); % find new pos x-,y- matched
uvr_cur_new = fr_cur(1:2,matches_new_v1(1,:));

% draw line across frames: h1=vertical(left), h2=vertical(right),
% h3=horizontal(left to right at current frames)
h1 = line([uvl_prev_new(1,:);uvl_cur_new(1,:)],
          [uvl_prev_new(2,:);uvl_cur_new(2,:)+size(limg_cur,1)]);
h2 = line([uvr_prev_new(1,:)+size(limg_cur,2);uvr_cur_new(1,:)+size(limg_cur,2)],
          [uvr_prev_new(2,:);uvr_cur_new(2,:)+size(limg_cur,1)]);
h3 = line([uvl_prev_new(1,:);uvr_prev_new(1,:)+ size(limg_prev,2)],
          [uvl_prev_new(2,:);uvr_prev_new(2,:)])
```

Figure 6. Code for finding correspondences between prev.-curr. frames with points matche left and right

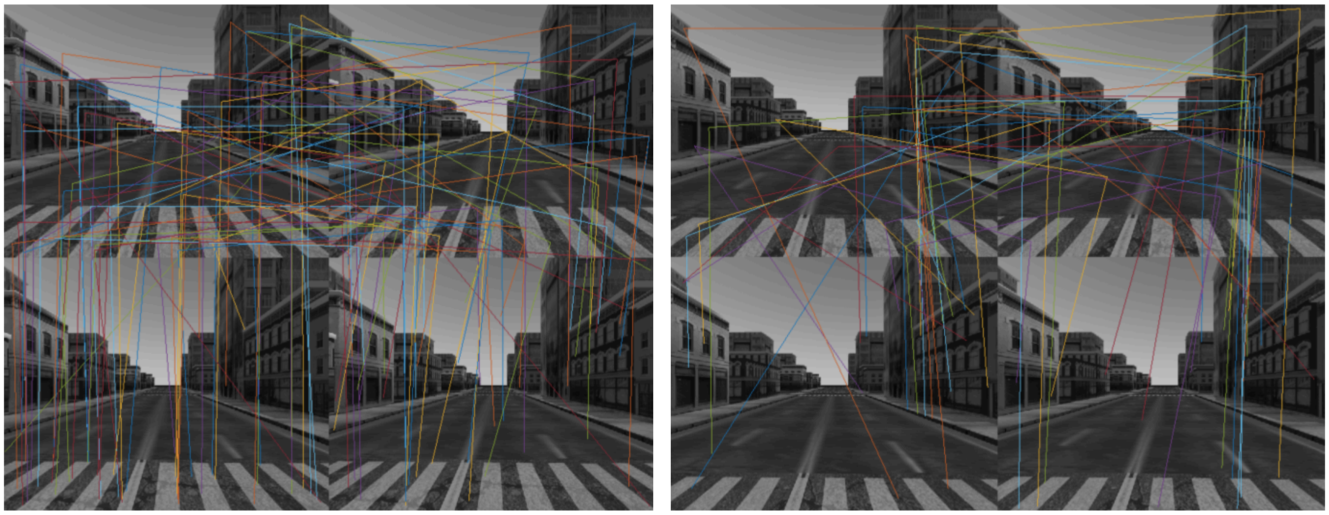


Figure 7. Correspondences drawn previous frame and current frame,
with points found commonly at left and right frame of current scene

5. Motion Estimation: given triangulated positions of landmarks from the previous frame and correspondences in current frame

(a) Write a residual function that predicts the measurement of the 3Dpoints in the current frame and computes the difference between tracked features locations (measurements)

```
    uvl = fl(1:2,matches(1,:));
    uvr = fr(1:2,matches(2,:));

    cmodl = struct('fx',268.5119, 'fy', 268.5119, 'cx', 320.5, 'cy', 240.5);
    cmodr = struct('fx',268.5119, 'fy', 268.5119, 'cx', 320.5, 'cy', 240.5);

    Twca = [ 1, 0, 0, 0.2; 0, 1, 0, 0.05; 0, 0, 1, -2.0 ];
    Twcb = [ 1, 0, 0, 0.2; 0, 1, 0, 0.75; 0, 0, 1, -2.0 ];

     [p, index] = triangulate(uvl_cur, uvr_cur, Twca, Twcb, cmodl,cmodr) % get 3d points
     [resL, resR] = residual(p, uvl, uvr) % get each of residual from 2d projection of 3d

function [resL, resR] = residual(p, uvl, uvr, Twca, Twcb, cmodl, cmodr)

    2dl = proj_3d_to_2d(Twca, p, cmodl);
    2dr = proj_3d_to_2d(Twcb, p, cmodr);

    resL = uvl(1:2,:) - 2dl(1:2,:);
    resR = uvr(1:2,:) - 2dr(1:2,:);
end

function [p, index_keep] = triangulate(uvl,uvr,Twl,Twr,lcmod,rcmod)

    % triangulation above
end
```

Figure 8. Residual function to measure difference between tracked features location

(b) RANSAC (Not solved)

i. Write a sub-routine [Tab,inliers] = RANSACMotion(z, m, p, sigma), that takes the tracked features, z, the corresponding 3D landmarks, m, the desired probability of finding the inliers, p, and the inlier standard deviation, sigma.

A. Compute the number of RANSAC iterations to find inlier model

B. Select a minimal subset of landmarks – how many are needed?
3 sets of subsets are needed

C. Compute the motion of the stereo cameras given the minimal subset

D. Score the computed motion model; count inliers; save the estimate

E. Return the motion estimate with the most number of inliers

(c) Using the inliers found Compute the 6DoF motion of the left camera

6. Visual Odometry
(a) Find the trajectory the camera followed; what is the final pose?
(b) Draw the full trajectory and plot it from above