Computer Vision Homework 2

Jeeeun Kim ([jeeeun.kim@colorado.edu](mailto:jeeeun.kim@colorado.edu))

**1. Write the matlab function uv = proj\_3d\_to\_2d( Twc, xwp, cmod ) where**

* **uv is the output 2D pixel image of 3D point xwp,**
* **xwp is the point p stored in world frame w,**
* **Twc is the 4x4 homogeneous pose matrix of camera c in world frame w,**
* **cmod is a matlab struct storing the camera intrinsics fx, fy, s, cx and cy– e.g., the 5 free terms of the camera calibration matrix, K. You should also store the image width and height in cmod, (e.g., as cmod.imagewidth and cmod.imageheight).**
* **Make sure the function can take a 3xN matrix of N 3D points in xwp (so one function call can project many points).**
* **Make sure to avoid projecting points that are behind the camera; also avoid projecting points that are outside of the image.**

|  |
| --- |
| function uv = proj\_3d\_to\_2d(twc, xwp, cmod)    K = [cmod.fx cmod.s cmod.cx;  0 cmod.fy cmod.cy;  0 0 1 ];    [r,c] = size(xwp);  xwpN = [xwp;ones(1,c)];    for i=1:c  xcp(1:4,i) = inv(twc)\*xwpN(1:4,i); %xcp = 4by4    if(xcp(3,i) > 0) %only for the positive-z  uvw(1:3,i) = K \* xcp(1:3,i);  uv(1:2,i) = [uvw(1:2,i)/uvw(3,i)];    hold on  xlim([0, cmod.imgW]); ylim([0, cmod.imgH]); %limit image plane  plot(uv(1,i),uv(2,i),'x');  disp([uv(1,i),uv(2,i)]); %display uv pairs on the console  end  end  end |

**Figure 1. function uv = proj\_3d\_to\_2d(twc, xwp, cmod)**

**2. Assume you are given the following camera model: fx=320, fy=240, s=0, cx=320, cy=240, image width=640, image height=480. Consider 9 3D points, arranged in a regular 3x3 grid with 20cm spacing, parallel to the xy-plane and 1m infront of the camera with the center point on the +z axis. Space the points 20cm apart.**

(a)  Write a test harness in matlab to project the 9 points into the camera; what are the image coordinates of the 9 points? Plot the image projection points and set the x-limits and y-limits according to the camera image size.

|  |
| --- |
| function test = test()    k = struct ('fx',320, 'fy',240, 's',0, 'cx',320, 'cy',240,  'imgW',640, 'imgH',480);  sp = 20;    xwp = [ -sp 0 +sp -sp 0 +sp -sp 0 +sp;  +sp +sp +sp 0 0 0 -sp -sp -sp;  100 100 100 100 100 100 100 100 100];  twc = cart2t([0 0 0 0 0 0]');  proj\_3d\_to\_2d(twc, xwp, k);  end |

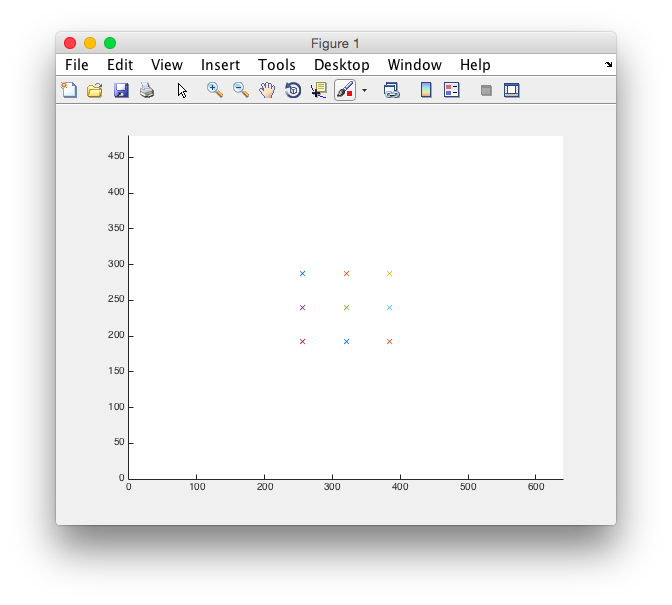
**Figure 2. Test harness with given camera calibration parameter and 9 points**

Given the pose = ([0 0 0 0 0 0]’), positions of uv is

uv =

256 320 384 256 320 384 256 320 384

288 288 288 240 240 240 192 192 192



**Figure 3. 2D projection of 9 3D points**

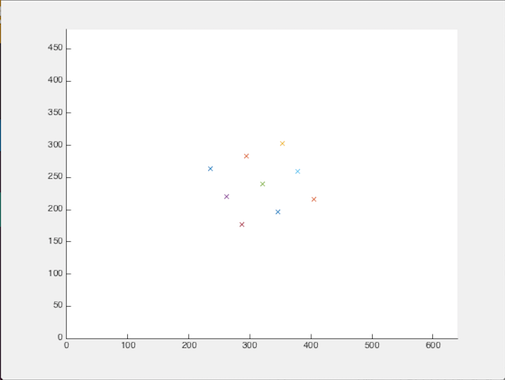
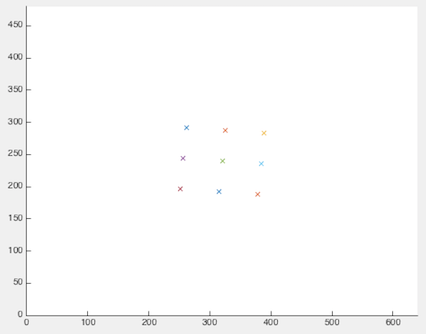
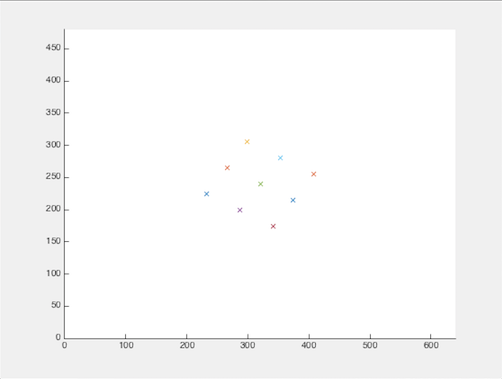
**(b)  In a loop for roll equals 0:0.1:8pi project the 9 points at each pose. Which direction do the poins on the screen move?**

**(c)  In a loop for pitch equals 0:0.1:8⇡projectthe9pointsateachpose.Which direction do the poins on the screen move?**

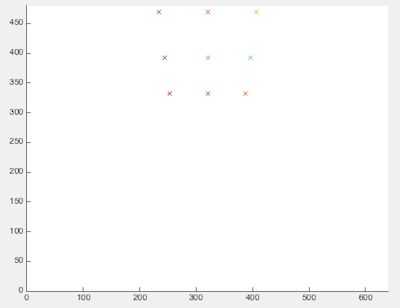
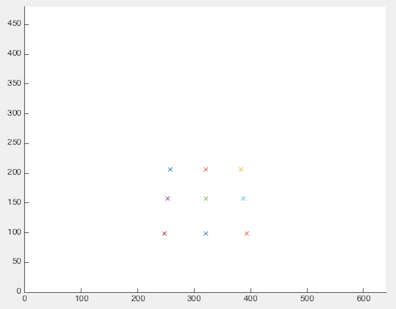
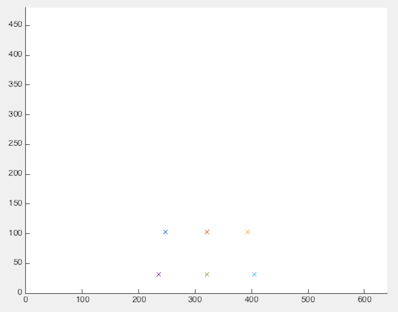
**(d)  In a loop for yaw equals 0:0.1:8pi project the 9 points at each pose.. Which direction do the points on the screen move?**

|  |
| --- |
| function test = test2()    k = struct ('fx',320, 'fy',240, 's',0, 'cx',320, 'cy',240, 'imgW',640, 'imgH',480);  sp = 20;    xwp = [ -sp 0 +sp -sp 0 +sp -sp 0 +sp;  +sp +sp +sp 0 0 0 -sp -sp -sp;  100 100 100 100 100 100 100 100 100];    % Question #2(b), roll  for p = 0:0.1:8\*pi %roll  twc = cart2t([0 0 0 p 0 0]'); %rotate in x  proj\_3d\_to\_2d(twc, xwp, k);  pause(0.02);  clf();  end    % Question #2(c), pitch  for q = 0:0.1:8\*pi %roll  twc = cart2t([0 0 0 0 q 0]'); %rotate in y  proj\_3d\_to\_2d(twc, xwp, k);  pause(0.02);  clf();  end    % Question #2(d), yaw  for r = 0:0.1:8\*pi %roll  twc = cart2t([0 0 0 0 0 r]'); %rotate in r  proj\_3d\_to\_2d(twc, xwp, k);  pause(0.02);  clf();  end  end |

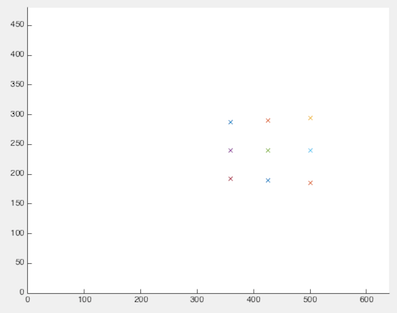
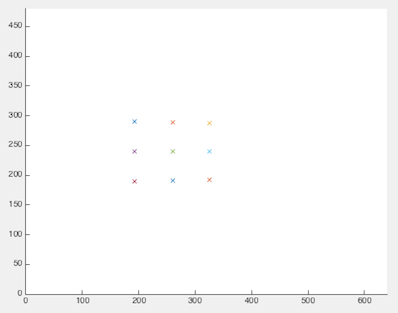
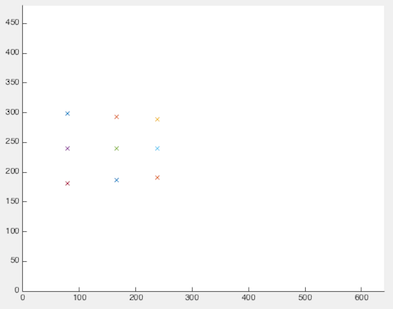
**Figure 4. Source code for question No.2b-e**

**Figure 5. Projection on roll, 2(b), rotating in a screen**

**Figure 6. Projection on pitch, 2(c), rotating from top to bottom**

**Figure 7. Projection on yaw, 2(d), rotating from right to left**

**(e)  For each pose, also compute the distance, d, from the camera center to each of the 9 points.**

For each pose, the distance from the camera to the distance does not change, since the points are circulating along with the points.

|  |
| --- |
| function d = dist(twc)  xyz = Twc(1:3,4);  d = sqrt((Xwp(1,:)-xyz(1)).^2+(Xwp(2,:)-xyz(2)).^2+(Xwp(3,:)-xyz(3)).^2);    end |

**Figure 8. Function d = dist(twc, xwp)**

**3. Write the matlab function ray = proj\_uv\_to\_3d( uv, cmod ) where ray is a unit ray in the camera frame.**

**(a) How would you transform the ray into the world frame?**

3D points measured from uv set are calculated based on the distance and ratio between the position of camera and the 2D frame where a set of uv situate.

|  |
| --- |
| function ray = proj\_uv\_to\_3d(uv, cmod)  d = 1; % unit distance  [r,c] = size(uv);  for i=1:c  u = (uv(1,i) - cmod.cx) / cmod.fx;  v = (uv(2,i) - cmod.cy) / cmod.fy;    D = sqrt(u^2 + v^2 + d^2);    ray(1,i) = u(1,i)/D;  ray(2,i) = v(1,i)/D;  ray(3,i) = d(1,i)/D;  end  end |

**Figure 8. Function ray = proj\_uv\_to\_3d(uv, cmod)**

4. Write the matlab function xwp = proj\_uvd\_to\_3d( uv, d, cmod ) where d is the depth along the pixel-ray in the camera frame.

|  |
| --- |
| function xwp = proj\_uvd\_to\_3d(uv, d, cmod)    [r,c] = size(uv);  ray = proj\_uv\_to\_3d(uv, cmod);  for i=1:c  xwp(1:3,i) = ray(1:3,i) \* d(1,i);  end  end |

**Figure 6. Function xwp = proj\_uvd\_to\_3d(uv, d, cmod)**

(a) Plug these into xwp = proj\_uvd\_to\_3d( uv, d, cmod ) and verify the results are correct. Use the distances, d, computed in problem 2e.