Name: Pham The Thinh
Student's ID: M10903821

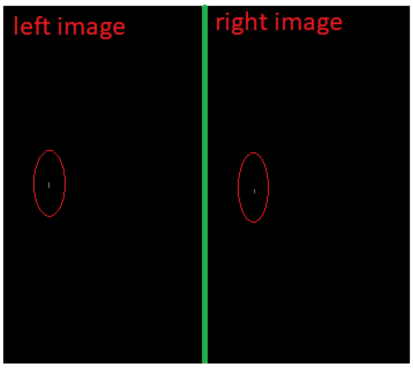
COMPUTER VISION AND APPLICATIONS

# FINAL PROJECT

## Reconstruct 3D from stereoscopic images and colorize 3D point cloud

### 1. Reconstruct 3D point.

- We will filter noise and remain the centerline extraction of both images as below picture.




- We are going to pick out each pixel coordinate of both images that lies in the centerline extraction and save to point left sets and point right sets.
- This is option 1, we took points of left image and draw the epipolar lines on right image to get the corresponding right points on the epipolar lines.
- This is option 2, we keep all right points and left points, we put them directly to the finding point pair function.
- We continuous to find the each pair of corresponding points $x \leftrightarrow x'$ in two point sets based on the fundamental matrix.

```
am = np.transpose(xyL[i])
# x'T * F * x : right image *fundamental matrix* left image
error = xyR[m].dot(F.dot(am))
```

- Before finding the 3D point, we need to find projective matrix of each camera by multiply intrinsic matrix and extrinsic matrix of each camera.

```
intrinsics_fundament = np.genfromtxt(our_data, skip_header=1,skip_footer=5)
extrinsics = np.genfromtxt(our_data, skip_header=10,skip_footer=2)
K_l = intrinsics_fundament[0:3]
K_r = intrinsics_fundament[3:6]
Rt_l = extrinsics[0:3]
Rt_r = extrinsics[3:6]
#############
P_L = K_l.dot(Rt_l)
print("P matrix of left camera",P_L)
P_R = K_r.dot(Rt_r)
print("P matrix of right camera",P_R)
F = intrinsics_fundament[6:9]
print("Fundamental matrix of both camera", F)
```

- From each pair, we will calculate 3D point respectively.

$$\begin{bmatrix} u\mathbf{p}_3^{\mathrm{T}} - \mathbf{p}_1^{\mathrm{T}} \\ v\mathbf{p}_3^{\mathrm{T}} - \mathbf{p}_2^{\mathrm{T}} \\ u'\mathbf{p}'^{\mathrm{T}}_3 - \mathbf{p}'^{\mathrm{T}}_1 \\ v'\mathbf{p}'^{\mathrm{T}}_3 - \mathbf{p}'^{\mathrm{T}}_2 \end{bmatrix} \mathbf{X}_{4\times1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
A = np.array(
[[mpr2[i][0] * P_r[2][0] - P_r[0][0], mpr2[i][0] * P_r[2][1] - P_r[0][1], mpr2[i][0] * P_r[2][2] - P_r[0][2],mpr2[i][0] * P_r[2][3] - P_r[0][3]],
[mpr2[i][1] * P_r[2][0] - P_r[1][0], mpr2[i][1] * P_r[2][1] - P_r[1][1], mpr2[i][1] * P_r[2][2] - P_r[1][2], mpr2[i][1] * P_r[2][3] - P_r[1][3]],
[mpl1[i][0] * P_l[2][0] - P_l[0][0], mpl1[i][0] * P_l[2][1] - P_l[0][1], mpl1[i][0] * P_l[2][2] - P_l[0][2], mpl1[i][0] * P_l[2][3] - P_l[0][3]],
[mpl1[i][1] * P_l[2][0] - P_l[1][0], mpl1[i][1] * P_l[2][1] - P_l[1][1], mpl1[i][1] * P_l[2][2] - P_l[1][2], mpl1[i][1] * P_l[2][3] - P_l[1][3]]])
USV = np.linalg.svd(A)
V = np.transpose(USV[2])
X = np.array([[V[0][3]], [V[1][3]], [V[2][3]], [V[3][3]]])
X = X / X[3][0]
points.append([np.round(X[0][0], 0), np.round(X[1][0], 0), np.round(X[2][0])])
```

## 2. Colorize 3D point.

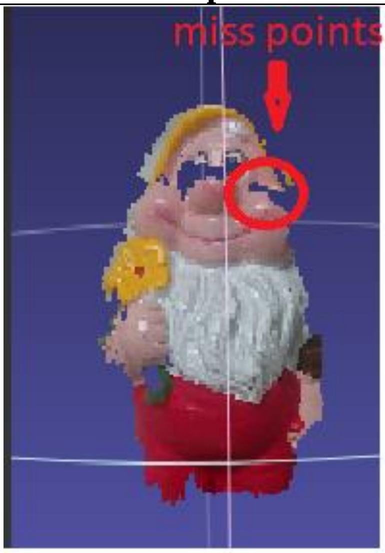
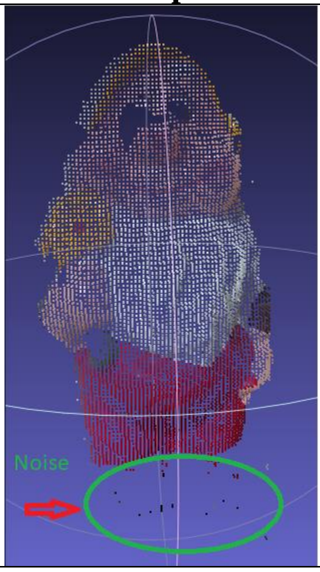
- We will find the projective matrix between 3D points and 2D image.
- Take at least 6 points to find the projective matrix

```
uv6 = [1562,1015,1];
% The matrix form to find transformation P
D = [XY1,0,0,0,0,-uv1(1).*XY1;
     0,0,0,0,XY1,-uv1(2).*XY1;
     XY2,0,0,0,0,-uv2(1).*XY2;
     0,0,0,0,XY2,-uv2(2).*XY2;
     XY3,0,0,0,0,-uv3(1).*XY3;
     0,0,0,0,XY3,-uv3(2).*XY3;
     XY4,0,0,0,0,-uv4(1).*XY4;
     0,0,0,0,XY4,-uv4(2).*XY4;
     XY5,0,0,0,0,-uv5(1).*XY5;
     0,0,0,0,XY5,-uv5(2).*XY5;
     XY6,0,0,0,0,-uv6(1).*XY6;
     0,0,0,0,XY6,-uv6(2).*XY6];
 % Calculate P by using SVD
[U,S,V] = svd(D);
P = [V(1:4,12)';V(5:8,12)';V(9:12,12)'];
% Normalized
P = P./P(3,4)
pp1 = P*XY1';
pp1 = pp1./pp1(3);
```

- Then, we compute all 2D coordinates from 3D points.
- Use it to get color position.

```
# Convert 3D point onto 2D image and get information uv on image of 3D points
P = np.array([[21.9034, -0.1663, 1.2928, 829.0357],
              [1.2204, 19.5062, -2.1049, 1592.4],
              [0.0012, 0.00005, -0.00009, 1]])
uv = P.dot(xyz_T)
uv = uv/uv[2]
# We will find color responding to position of each 3D points on image
color = texture[int(np.round(uv[1],0)),int(np.round(uv[0],0))] # (B,G.R) and [y,x]==[v,u]
colors.append([color[2],color[1],color[0]]) # R,G,B
```

**The results:**

| **This is option 1** | **This is option 2** |
|---|---|
|  |  |

Comments:

- The result of the one option can remove noise and outliers but it is missed some points. Meanwhile the result of the option 2 have little missed point than option 1 and there are more outliers.

- The colorized 3D point is not good, because choosing corresponding points between 2d and 3d point is not good.