Project 2 - 3D Reconstruction from two views

Daniel Bryan and Olmo Zavala
Florida State University

November 12, 2014

1 Description

The objective of this project is to implement the normalized eight-point algorithm for estimating the fundamental matrix, and how to generate 3D models from images.

2 Eight-point algorithm

Our implementation of the 8-pt algorithm is based on the slides of the class and it basically follows the following algorithm:

- Normalize image points
  - **Centroid is at the origin**. We create the matrix T_{trans} for each camera like this:
$$\begin{bmatrix} 1 & 0 & -\mu_x \\ 0 & 1 & -\mu_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$
    And we multiply each point of the cameras to they corresponding T matrix like this: Tx_i.
  - **RMS distance from the origin is $\sqrt{2}$.** First compute the RMS of the available points:
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}((x_i-\mu_x)^2+(y_i-\mu_y)^2))} \tag{2}$$
    Then create T_{scale} and multiply it to each point in the camera. T is:
$$T_s = \begin{bmatrix} \sqrt{2}/RMS & 0 & 0 \\ 0 & \sqrt{2}/RMS & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$
- **Multiply each point by $T_n = T_sT_t$ like this $[u\ v\ 1]' = T_nx$. Do it for each camera.**
- **Solve** $x'_nFx_n = 0$. To do this we need to form the system $Af = 0$ and solve for f. The matrix A is:
$$A = \begin{bmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2u_2 & u'_2v_2 & u'_2 & v'_2u_2 & v'_2v_2 & v'_2 & u_2 & v_2 & 1 \\ u'_3u_3 & u'_3v_3 & u'_3 & v'_3u_3 & v'_3v_3 & v'_3 & u_3 & v_3 & 1 \\ & & & & \vdots & & & & \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{bmatrix} F = 0 \tag{4}$$

- **Find least square solution of $Af = 0$.**
  - First find SVD of A ($USV^T = A$).
  - Choose F_{Norm} to be the last column of V
- **Enforcing Singularity**
  - For $F_{Norm} = USV^T$
  - Set S_3=0 for

$$F_{norm} = U \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{bmatrix} V^T \tag{5}$$

- **Denormalisation**

$$F = T_{NormLeft} * F_{Norm} * T_{NormRight} \tag{6}$$

- **Compute essential matrix E**

$$E = K^T * F * K \tag{7}$$

  Where K are the intrinsic camera parameters of the Kinect camera:

$$K = \begin{bmatrix} -525 & 0 & 320 \\ 0 & -525 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

- **Estimate R and T from E**
  - SVD on E. $E = USV^T$
  - $R = UWV^T$ or $R = UW^TV^T$. Where W i given by

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

  - T can be u_3 or $-u_3$.
- **Choose proper solution**. Finally we need to choose from the four possible solutions the correct one.
  - For each case build matrix A

$$A = \begin{bmatrix} -1 & 0 & x_1 & 0 \\ 0 & -1 & y_1 & 0 \\ -R_{11} + x_2R_{31} & -R_{12} + x_2R_{32} & -R_{13} + x_2R_{33} & -T_1 + x_2T_3 \\ -R_{21} + y_2R_{31} & -R_{22} + y_2R_{32} & -R_{23} + y_2R_{33} & -T_2 + y_2T_3 \end{bmatrix} \tag{10}$$

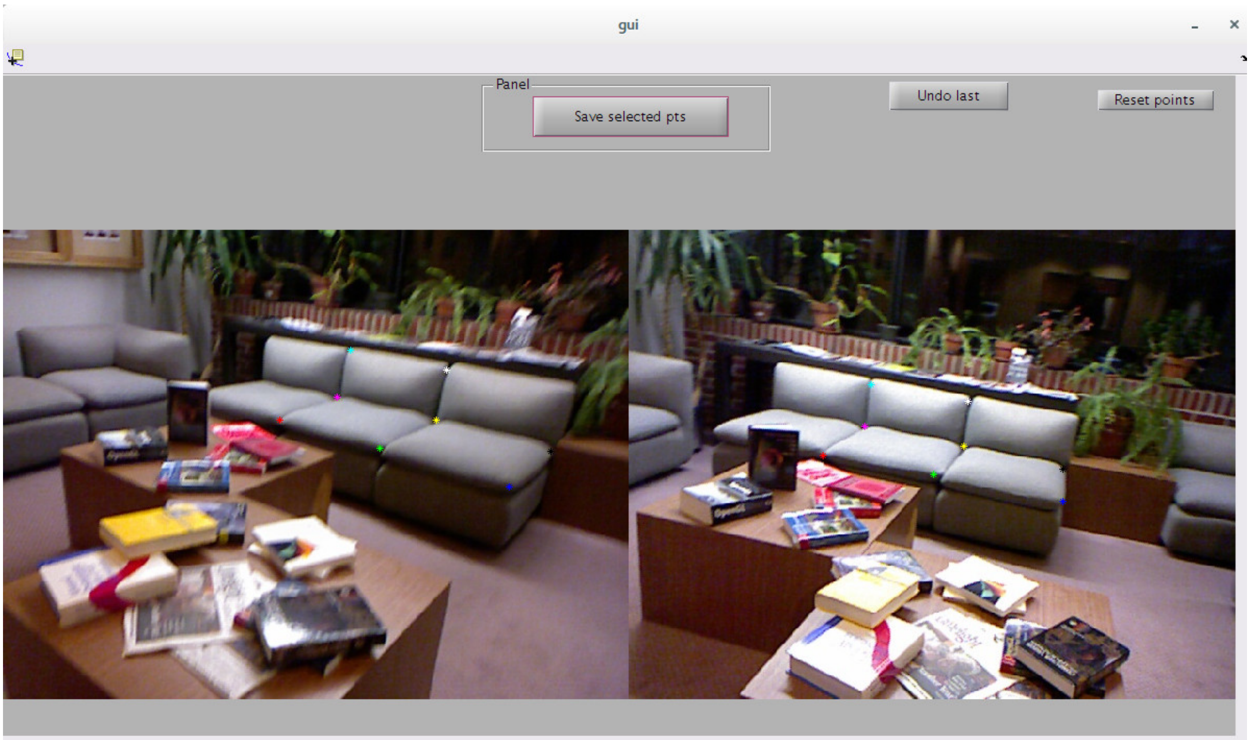

Figure 1: Matlab interface to select corresponding points between two images. Example of selecting 9 points using the Matlab interface.

3 Interface to define corresponding points.

We created a useful Matlab interface to define the corresponding points between two images. Figure 1 shows a screen shot of this interface. The user needs to select corresponding points between images and it doesn't have to be in order (one point in one image and the next point in the following image). The interface also provides a button to reset all the points as well as to remove the last added point. These two buttons are very handy when selecting points. Once all the desired buttons have been selected the resulting points are saved in at matlab file that can be stored for later compuatation using the 8-point algorithm.

Each corresponding point is displayed with the same color, this makes it easy to see wich points correspond in both of the images. The coordinates obtained from the selection of points displayed at figure 1 are displayed in figure 2.

```
x_l =

  283.7333  285.3626
  386.1333  255.8242
  518.4000  216.7912
  561.0667  252.6593
  443.7333  284.3077
  342.4000  308.5714
  356.2667  356.0440
  454.4000  336.0000


x_r =

  199.4667  248.4396
  312.5333  229.4505
  444.8000  202.0220
  444.8000  234.7253
  343.4667  257.9341
  242.1333  277.9780
  248.5333  321.2308
  347.7333  304.3516
```

Figure 2: Obtained coordinates from the selected in figure 1.

In our results we use the example provided by the professor to compare the results.

In order to test the correct coordinates of the points obtained by the Matlab interface we selected 4-points close to the corners of the two figures as displayed in figure 3. The points start in the upper right corner and go clockwise until the upper left corner. The values obtained correspond very closely with the dimensions of the images but with the origin in the lower left corner. The values obtained are also displayed in figure 3.

```
x_l =

  634.6667  468.9231
  636.8000    5.8022
    6.4000    8.9670
    4.2667  472.0879

x_r =

  617.6000  469.9780
  617.6000    6.8571
    6.4000    6.8571
    5.3333  465.7582
```

Figure 3: Corners used to test the proper coordinates of the Matlab interface.

4 Results

To test our results we use the example provided by the profesor. This example were 12 points with the coordinates shown in figure 4.

```
x_l = [           x_r = [
    434.00 360        428 354
    468.00 542        447 545
    275.00 393        255 389
    297.00 542        265 545
    400.00 484        407 481
    424.00 625        421 629
    242.00 483        239 481
    261.00 628        248 631
    631.00 541        625 544
    655.00 622        671 627
    787.00 540        797 544
    650.00 443];      668 439];
```

Figure 4: Coordinates used to test our results.