Project: Stereo View Camera Rectification
CIS 580, Machine Perception, Spring 2016
Due: 2016.05.10. 11:59AM

This project provides an in-depth understanding of how to rectify a pair of stereo images, to bring the images into their ideal stereo view configuration. This is achieved by computing a desired Homography mapping. We assume the cameras are calibrated, with known intrinsic parameters and external relative camera pose.

In section 1 to 3, you’ll finish three Matlab functions for homography transformation construction, image warping and epipole estimation. In section 4, you’ll finish a Matlab function to decompose rotation operation, and then use previous functions to 1) rotate the camera image, 2) visually experience the camera rotation by warping images, 3) relate camera rotation to epipolar line and epipole changes, and 4) finally finish stereo rectification.

1 Homography transformation

In this section, you will construct homography transformation between image planes with respect to camera rotation.

**Goal**
Given camera intrinsic parameter \( K \) and rotation \( R_c \) between two images, construct homography transformation \( H \) that maps points on image 2 (after camera rotation) to image 1 (before camera rotation), i.e., \( \lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ 1 \end{bmatrix} \), where \( x_1 \) is the corresponding point on image 1 (before camera rotation) and \( x_2 \) is a point on image 2 (after camera rotation):

\[
[H] = \text{EstimateHomography}(K, R_c)
\]

(INPUT) \( K \): camera intrinsic parameters

(INPUT) \( R_c \): camera rotation rotation matrix
Recall Problem 2.2 in homework 2, the projection equation before and after the camera rotation is
\[ \lambda \begin{bmatrix} x_1' \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} \] and \[ \lambda \begin{bmatrix} x_2' \\ 1 \end{bmatrix} = \mathbf{K} \mathbf{R}_c \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} , \] respectively. Thus, \[ \lambda \begin{bmatrix} x_1' \\ 1 \end{bmatrix} = \mathbf{K} \mathbf{R}_c^{-1} \mathbf{K}^{-1} \begin{bmatrix} x_2' \\ 1 \end{bmatrix}, \] and \[ \mathbf{H} = \mathbf{K} \mathbf{R}_c^{-1} \mathbf{K}^{-1}. \]

### 2 Image Warping

In this section, you will warp the original image to generate the image after camera rotation.

**Goal** Given homography transformation \( \mathbf{H} \) between two images, warp original image to generate new image:

\[ \text{[imWarp]} = \text{WarpImage(im, H)} \]

(INPUT) \( \text{im} \): original image with size \( H \times W \times 3 \), where \( H \) and \( W \) are the height and width of the image

(INPUT) \( \mathbf{H} \): homography transformation which maps the points from \( \text{imWarp} \) to \( \text{im} \)

(OUTPUT) \( \text{imWarp} \): warped image with size \( H \times W \times 3 \).

For each point \( \mathbf{x} \) in warped image, coordinate for the corresponding point \( \mathbf{x}' \) on original image is
\[ \lambda \begin{bmatrix} x_1' \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_1 \\ 1 \end{bmatrix} \] and its color should the the same as \( \mathbf{x}' \) in original image. Use interpolation (interp2 in Matlab) to get RGB value for non-integer \( \mathbf{x}' \).

### 3 Epipole Estimation

In this section, you will use intrinsic camera parameters and relative camera pose to build fundamental matrix and estimate epipoles on images.

**Goal** Given relative camera pose \( \{ \mathbf{R}_r, \mathbf{t}_r \} \) between two cameras, i.e., external parameters for left camera is \( \{ \mathbf{I}_3, \mathbf{0} \} \) (\( \mathbf{P}_l = \mathbf{K} \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} \end{bmatrix} \)) and external parameters for camera right is \( \{ \mathbf{R}_r, \mathbf{t}_r \} \) (\( \mathbf{P}_r = \mathbf{K} \begin{bmatrix} \mathbf{R}_r & \mathbf{t}_r \end{bmatrix} \)), compute the coordinates of epipole on both images:

\[ [\mathbf{e}_1, \mathbf{e}_2, \mathbf{F}] = \text{ComputeEpipole(K, R, t)} \]

(INPUT) \( \mathbf{K} \): intrinsic camera parameters \( \mathbf{K} \) for both camera

(INPUT) \( \mathbf{R} \): rotation matrix \( \mathbf{R}_r \) for right camera

(INPUT) \( \mathbf{t} \): translation \( \mathbf{t}_r \) for right camera

(OUTPUT) \( \mathbf{e}_1 \): epipole coordinate on left image (projection of right camera on left image).

(OUTPUT) \( \mathbf{e}_2 \): epipole coordinate on right image (projection of left camera on right image).

(OUTPUT) \( \mathbf{F} \): fundamental matrix between two images.

Recall in epipolar geometry, fundamental matrix is given by \( \mathbf{F} = \mathbf{K}^{-T} \mathbf{RT_xK}^{-1} \) and relates the corresponding points on two images by \( \begin{bmatrix} \mathbf{x}_r^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_l \end{bmatrix}^T \mathbf{F} \begin{bmatrix} \mathbf{x}_l \end{bmatrix} = 0 \). Epipole is the projection of one camera center onto the other camera’s image plane, and \( \begin{bmatrix} \mathbf{e}_2^T \end{bmatrix} \mathbf{F} = 0, \begin{bmatrix} \mathbf{e}_1 \end{bmatrix}^T \mathbf{F} = 0 \).

The camera intrinsic parameters \( \mathbf{K} \), camera pose of the right camera \( \mathbf{R}_r, \mathbf{t}_r \) are given in camera-Pose.mat. Correspondences between the two images \( \{ \mathbf{x}_l \leftrightarrow \mathbf{x}_r \} \) are given in correspondence.mat.

Use the relative camera pose to compute the coordinates of epipole on both images, and verify your answer by checking the intersection of epipolar lines.
4 Rectification

In this section, you will rectify a pair of images by two steps: (1) rotate the right camera to make its orientation parallel to the first camera and (2) rotate two cameras together to make their z-axis orientation perpendicular to their baseline. In the process, you will need to generate new images to experience camera rotation and visualize the epipolar line and epipole in both images to get an intuitive understanding of how epipolar geometry is related to camera rotation.

Compelete given matlab script Rectification to finish the whole pipeline and generate output video.

4.1 Decompose Rotation Matrix

**Goal** Decompose a rotation operation denoted by $R$ into the concatenation of $n$ micro rotation operation denoted by $R_\delta$, i.e., $R = R_\delta^n$:

$$[R_{\delta}] = \text{DecomposeR}(R, n)$$

(INPUT) $R$: rotation matrix for original rotation operation

(INPUT) $n$: number of micro rotation operation

(OUTPUT) $R_{\delta}$: rotation matrix for micro rotation operation.

4.2 Rotate Right Camera for Parallel Orientation

Rotate the right camera so that its extrinsic parameters change from $\{R_r, t_r\}$ to $\{I_3, -R^T_r t_r\}$.

To experience the rotation process from image aspect, decompose the rotation of the right camera into 10 steps and rotate the right camera by $R_\delta$ for each step, $R_{10}^\delta = R^T_\delta$. To be more explicitly, in each step of applying $R_\delta$ to right camera:

1. Use `EstimateHomography` to construct homography transformation $H$ for mapping images before and after camera rotation.

2. Use `WarpImage` to warp right image by homography transformation $H$ to generate new right image after applying $R_\delta$ to right camera.

3. Use `ComputeEpipole` to compute epipole on both images and draw epipolar lines to verify your epipole position by the given correspondences.

Hint: When the right camera is rotated to have the same orientation as the left camera, the coordinates for epipole on two images should be the same.

Figure 2: Rotate Right Camera for Parallel Orientation
4.3 Rotate Both Camera for Rectification

Having the same orientation of both cameras, then rotate both camera together to make their z-axis orientation (last row of the rotation matrix \( R \)) perpendicular to their baseline \(-R_t_r\). Similarly, decompose the rotation of the cameras into 10 steps to experience the change of epipolar lines and epipoles. For each rotation operation:

1. Use \textit{EstimateHomography} to construct homography transformation \( H \) for mapping images before and after camera rotation.

2. Use \textit{WarpImage} to warp left image by homography transformation \( H \) to generate new left image after applying \( R_\delta \) to left camera.

3. Use \textit{WarpImage} to warp right image by homography transformation \( H \) to generate new right image after applying \( R_\delta \) to right camera.

4. Use \textit{ComputeEpipole} to compute epipole on both images and draw epipolar lines to verify your epipole position by the given correspondence.

\textbf{Hint:} When the cameras are rotated so that their z-axis orientation are perpendicular to their baseline (rectification is done), the coordinates for epipole on both images should be at infinity and the epipolar lines should be parallel.

![Image of rotated camera views](image.png)

\textit{Figure 3: Rotate Both Camera for Rectification}

5 Submission

Please submit following files:


2. Matlab script \textit{Rectification} to run the whole rectification pipeline.

3. Rotation video for two steps of rectification with epipolar lines drawn (use code given in \textit{Rectification} to generate videos).