**Assignment 4**
Computer Vision 15–385, Spring 12
Due Date: Tuesday 04/12/2012
Total Points: 90

In this assignment, you will be implementing the rectification and the depth estimation from stereo image pairs, given camera calibration parameters.

# 1 Submitting Your Assignment

Your submission for this assignment will comprise of answers to a few theory questions, the code for your MATLAB implementation and a short writeup describing any thresholds and parameters used, interesting observations you made or things you did differently while implementing the assignment. The answers to the theory questions in Section 2 should be in a plaintext file or a pdf named `theory.txt` or `theory.pdf`. Each of the MATLAB functions you write as described in Section 3 along with any extra helper functions should be in a folder named `matlab`, upload only `.m` files to this folder and make sure all the files needed for your code to run (except data) are included. Also, you need to submit some `.mat` files and images files to show your intermediate and final results. Please refer to Section 3 and Section 4 . A directory has been created on for uploading all your course related files. The directory can be found at `/afs/cs.cmu.edu/academic/class/15385-s12-users/`*andrewid* (for graduate students, the folder is `15685-s12-users`. Inside your submission directory, create a subfolder named `p4` for this assignment. Do not add any extra layers of indirection to your directory structure. Your final upload should have the files arranged in this layout:

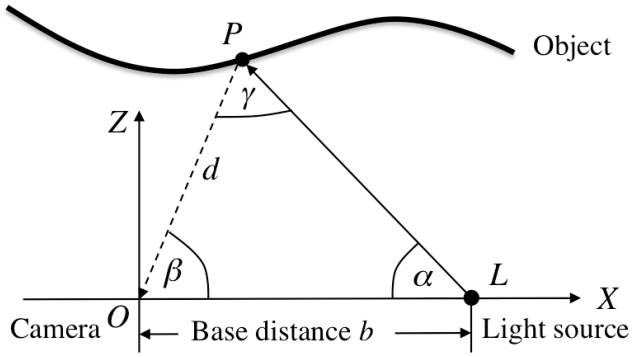
- `/afs/cs.cmu.edu/academic/class/15385-s12-users/`*andrewid*
  - `p4`
    - `theory.txt` or `theory.pdf`
    - `result`
      - `rect_0_office.png` and `rect_0_street.png`
      - `rect_office.mat` and `rect_street.mat`
      - `disparity_0_office.png` and `disparity_0_street.png`
      - `depth_0_office.png` and `depth_0_street.png`
      - `depth_office.zip` and `depth_street.zip`
    - `matlab`
      - `rectify_pair.m`
      - `q4rectify.m`, `warp_stereo.m`, and `p2t.m` (*already provided*)
      - `get_disparity.m` and `get_depth.m`
      - `q4depth.m` (*already provided*)

- `q4stereo.m`

You may need to run `aklog cs.cmu.edu` when you log in to be able to read and write from your submission directory. Your files are due by 23:59:59 on the submission day. Please make sure you have write permissions to your submission folder ahead of time so that problems (if any) can be fixed. We will be using timestamps to determine submission times and for late day counting, so do not modify your files after the submission deadline unless you wish to use a late day.

# 2 Theory Questions

**Question 1: Triangulation for Structured Light in 2-D** (8 points)




Suppose that by calibration you already know the base distance $b$ and angles $\alpha$ and $\beta$. Using the fact that the point in object $P$, the camera center $O$, and the light source $L$ define a triangle, compute the $(x, z)$ coordinates of $P$ with respect to the camera center $O$ (i.e., $P$ should be represented by only $b$, $\alpha$, and $\beta$).

(*Hint*) You many need to use the law of sines.

**Question 2: Rectification** (12 points)

Suppose that you performed the rectification for a stereo rig composed by two pinhole cameras. For each of the following questions, respond with true or false and justify your answer in one sentence.

(a) Given a point $m_L$ in the left image, its epipolar line in the right image, the optical center of the right camera, and $m_L$ lie in the same plane.

(b) If the optical center of the right camera is in the focal plane of the left camera, then the left epipole lies inside the left image.

(c) Let $x_L$ and $x_R$ be the projected image locations of a 3D point $X$ in left and right images. If we know the positions of $x_L$ and left/right epipoles $e_L$ and $e_R$, then we can uniquely decide the position $x_R$.

(d) After rectification, the camera pair must share the same focal plane (*i.e.* the plane parallel to the image plane containing the optical center).

# 3 Programming

We provide the following data files in `hw4.zip`.

- `office`/`street`: Each folder includes 10 pairs of stereo images sampled from two video clips. The format of file names is `img_(frame no)_(c1/c2)_*.pgm` where `c1` is a left camera image and `c2` is a right camera image.
- `camera_office.mat`/`camera_street.mat`: Each file includes camera parameters (`A1`, `A2`, `R1`, `R2`, `T1`, `T2`) for office/street datasets. 1 and 2 indicate the left and right cameras, respectively. `A`, `R`, and `T` are the intrinsic parameter, the rotation matrix, and the translation vector, respectively. We use this convention throughout this homework.
- `gt_office.mat`/`gt_street.mat`: Each file includes groundtruth correspondences. You will use these files only to verify your results.

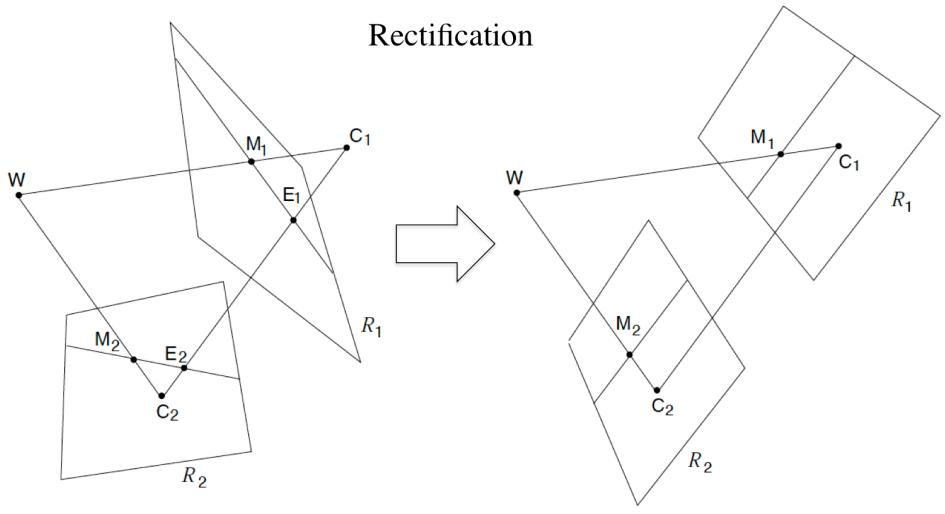
## 3.1 Rectification given calibration parameters.

**Part 1** Rectification matrices (20 points)

Write a program that computes rectification matrices.

```
function [M1,M2,A1n,A2n,R1n,R2n,T1n,T2n] = rectify_pair (A1,A2,R1, R2,T1,T2)
```

This function takes left and right camera parameters (`A,R,T`) and returns left and right rectification matrices (`M1,M2`) and updated camera parameters. You can implement this function using the provided script `q4rectify.m` (see **Part 2**).




Each camera is modeled by the *perspective projection matrix* (PPM) or *camera matrix* $P$ where $P = A[R|t]$. As shown in the figure, the basic idea of rectification is to obtain new PPMs $\widetilde{P}_1$ and $\widetilde{P}_2$ by rotating the two focal planes so as to be coplanar without moving the

optical centers ($c_1$ and $c_2$). In addition, any pair of corresponding points must have the same vertical coordinates. These constraints lead that two new PPMs can be thought as a single camera translated along the $X$ axis of its reference frame. Therefore, two new PPMs will have the same intrinsic and rotation matrices and only differ in their optical centers.

$$\widetilde{P}_1 = \widetilde{A}[\widetilde{R}|\widetilde{t}_1], \quad \widetilde{P}_2 = \widetilde{A}[\widetilde{R}|\widetilde{t}_2] \quad \text{where } t_1 = -\widetilde{R}c_1, \;\; t2 = -\widetilde{R}c_2 \tag{1}$$

The `rectify_pair` function should consecutively run the following steps.

1. Compute the optical center $c_1$ and $c_2$ of each camera. Since the optical centers do not move during rectification, we can compute it by using old PPMs. If we let $P = A[R|t] = [Q|q]$ (i.e., $Q = AR, q = At$), then $c = -Q^{-1}q$.
2. Compute the new rotation matrix $\widetilde{R} = [r_1 \;\; r_2 \;\; r_3]^T$ where $r_1$, $r_2$, and $r_3$ are $3 \times 1$ orthonormal vectors that represent $X$, $Y$, and $Z$ axes of the camera reference frame, respectively.
   (1) The new $X$ axis ($r_1$) is parallel to the baseline: $r_1 = (c_1 - c_2)/\|c_1 - c_2\|$.
   (2) The new $Y$ axis ($r_2$) is orthogonal to $X$ and to any arbitrary unit vector $k$. We set $k$ to be the $Z$ unit vector of the old left matrix: $r_2$ is the cross product of $R_1(3,:)^T$ and $r_1$.
   (3) The new $Z$ axis ($r_3$) is orthogonal to $X$ and $Y$: $r_3$ is the cross product of $r_1$ and $r_2$.
3. Compute the new intrinsic parameter $\widetilde{A}$. We can use an arbitrary one. In our test code, we just let $\widetilde{A} = A_2$.
4. Now we can compute $\widetilde{P}_1$ and $\widetilde{P}_2$ by plugging $\widetilde{R}, \widetilde{A}, c_1$, and $c_2$ into Eq.(1).
5. Finally, the rectification matrix of the first camera can be obtained by $M_1 = \widetilde{Q}_1 Q_1^{-1}$ if we let $P_1 = [Q_1|q_1]$ and $\widetilde{P}_1 = [\widetilde{Q}_1|\widetilde{q}_1]$. $M_2$ can be computed from the same formula.

In sum, The `rectify_pair` function takes $P_1 = A_1[R_1|t_1]$ and $P_2 = A_2[R_2|t_2]$ as inputs, and returns $M_1$, $M_2$, $\widetilde{P}_1 = \widetilde{A}_1[\widetilde{R}_1|\widetilde{t}_1]$, and $\widetilde{P}_2 = \widetilde{A}_2[\widetilde{R}_2|\widetilde{t}_2]$.

**Part 2** Testing your rectification matrices (10 points)

We provide the script called `q4rectify.m` so that you can test your `rectify_pair` function. This code will generate one mat file and one png file. The mat file saves the rectification parameters (i.e., the outputs of `rectify_pair` function), and the png file shows the rectified image pair.

## 3.2 Disparity and Depth Estimation

**Part 3** Disparity map (10 points)

Write a program that creates a disparity map from a pair of rectified images (`IL` and `IR`).

```
function dispM = get_disparity(IL,IR,maxDisp,windowSize)
```

where maxDisp is the maximum disparity and windowSize is the window size. The output dispM has the same dimension as IL and IR. Since IL and IR are rectified, computing correspondences is reduced to a 1-D search problem.

The $\texttt{dispM}(x, y)$ is

$$\texttt{dispM}(x, y) = \underset{0 \le d \le \texttt{maxDisp}}{\arg\min} \; d(\texttt{IL}(x, y), \texttt{IR}(x - d, y))$$

where $d(\texttt{IL}(x, y), \texttt{IR}(x-d, y)) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} (\texttt{IL}(x+i, y+j) - \texttt{IR}(x+i-d, y+j))^2$ with $w$ is $(\texttt{windowSize} - 1)/2$. This summation on the window can be easily computed by using the conv2 Matlab function (i.e. convolve with a mask of ones(windowSize,windowSize)) Note that this is not the only way to implement this.

(*Constraint*) For real-time applications, computing disparity map is usually required to be very fast. Here we encourage you to minimize the use of for loops. The full credit will be given only if your function uses a single for i=0:maxDisp loop.

**Part 4** Depth map (5 points)

Write a program that creates a depth map from a disparity map (dispM).

```
function depthM = get_depth(dispM,A1,A2,R1,R2,t1,t2)
```

Use the fact that $\texttt{depthM}(x, y) = b \times f / \texttt{dispM}(x, y)$ where $b$ is the baseline and $f$ is the focal length of camera. For simplicity, assume that $b = \|c_1 - c_2\|$ (i.e., distance between optical centers) and $f = A_1(1, 1)$. Finally, let $\texttt{depthM}(x, y) = 0$ at $\texttt{dispM}(x, y) = 0$.

**Part 5** Testing disparity map and depth map (10 points)

We provide the script called q4depth.m so that you can test your disparity and depth map, each of which will be saved as a png file. In order to run q4depth.m, you first complete **Part 2** because it requires the outputs of q4rectify.m.

Run q4depth.m for office and street dataset, and submit mat and png output files (disparity_office.mat, depth_0_office.png, disparity_street.mat, and depth_0_street.png).

# 4 Experiments

(15 points)

Write a program that performs rectification and estimates depth/disparity maps for the all images in the specified folder dirIn. fParam is a camera parameter file (ex. camera_office.mat or camera_street.mat). You can freely re-use q4rectify.m and q4depth.m.

```
function q4stereo(dirIn, fParam)
```

This function creates an output folder ['output_' dirIn] and saves four output files per one stereo pair: (1) the rectified image pair with the prefix of rec1_ and rec2_, (2) the disparity map with the prefix of disp_, and (3) the depth map with the prefix of dept_.

Use the name of the left image as the base file name. Please **resize all output images as [240×320]** before saving (use the Matlab function `imresize`).

Run the program with both office and street dataset (e.g. `q4stereo('office','camera_office.mat');` and `q4stereo('street','camera_street.mat');`). **Zip only the depth map results** as `depth_office.zip` and `depth_street.zip`, and submit them.