Camera models and stereovision

1 Introduction

The objective of this session is to practice with camera models and stereo-vision, following the lecture 4 of Computer Vision course. You can get more information about lectures and associated practical sessions on my website: `http://chateaut.fr`. Essential needed knowledges includes a beginning level of computer science (Linux, Python programming with `Opencv`, `Numpy`, `Matplotlib`, and `Pytorch` (for sessions on Deep Learning)) and applied mathematics.

The listing 1 shows an "Hello Word" python script that loads and convert an image using `opencv`; and displays it with `matplotlib` and illustrates relevant functions.

2 Camera Projection Matrix

Figures 1 and2 show the configuration of a virtual stereo system that looks at a cube. The Word reference R_w is set at the center of the vector between the optical axis of the two cameras. The orientation of each camera is 20^o from the z axis. The optical center of the cameras are at 2 meters from the world reference frame.

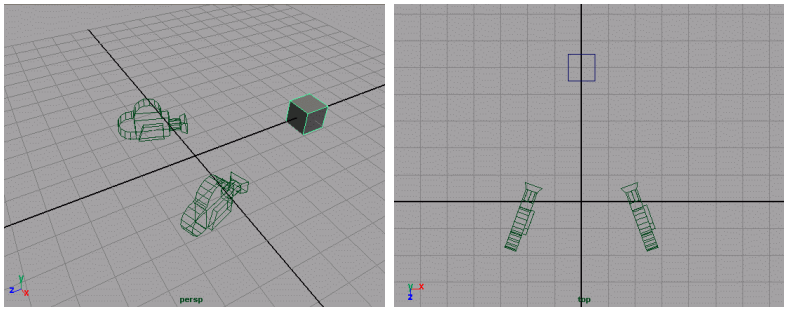

Figure 1: Left: 3D projection of the vision system; right: top view

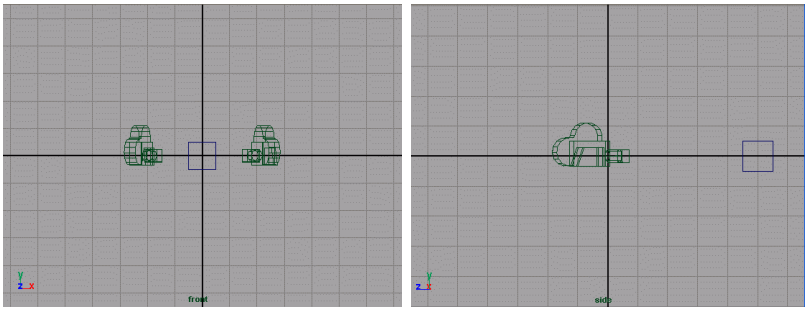

Figure 2: Left: front view; right: right view

Intrinsec parameters of the two cameras are:

- $f = 1750$pixels
- $u0 = 800$
- $v0 = 600$
- No distortion

Figure 3 shows the image acquired from the left camera. The python script `test_geom.py` is a baseline for the following questions.

Questions:

1. Compute the extrinsic matrices (homogeneous transformation matrix between the world reference frame and the cameras reference frames).
2. build the intrinsic matrix
3. Compute the projection model for the two cameras $C1$ (for the left camera) and $C2$ (for the right one). **TIP: the matrix product operator in python is `@`**
4. Project some corners on the images to validate the models.

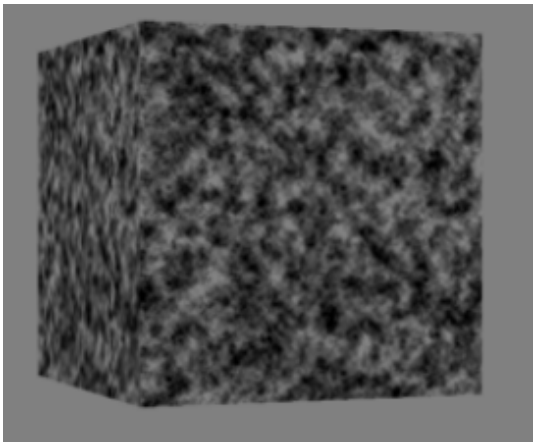

Figure 3: Left image: view of the cube

3 stereo-vision

Given the projection matrices of the two cameras ($C1$ and $C2$), and the projection of the same 3D point $\mathbf{P} = (P_x, P_y, P_z)^T$ in the two cameras ($\mathbf{p_1} = (u_1, v_1)^T$ and $\mathbf{p_2} = (u_2, v_2)^T$), the aim of this section is to estimate the coordinates of the 3D point $\mathbf{P}$ from $\mathbf{p_1}$ and $\mathbf{p_2}$.We will notice $\tilde{\mathbf{P}} = (\mathbf{P}^T, 1)^T$ the homogeneous vector associated to $\mathbf{P}$.

1. compute the relation between $\tilde{\mathbf{p_x}}$ and $\tilde{\mathbf{P_x}}$
2. deduce the equation betwen u_x and $(P_x, P_y, P_z)^T$

3. write, as a linear system ($\mathbf{AP} = \mathbf{B}$) the 4 equations that link $\mathbf{P}$ and $\mathbf{p_1}, \mathbf{p_2}$
4. propose an analytic solution to the linear system $\mathbf{AP} = \mathbf{B}$ (that minimizes a least square error)
5. You should complete the The python script `test_geom2.py` to write the linear system and solve it. The script ask for a manuel selection of one point in bith images and compute the coordinates of the 3D estimated point.
6. modify the script to display the projection of the 3D point on both images.

4 Application to 3D measure of distances in stereo-images

The aim of the section is to estimate 3D distances from a couple of stereo images. The figure 4 shows a pair of stereo images from the dataset Kitti. The script `test_geom_kitti.py`

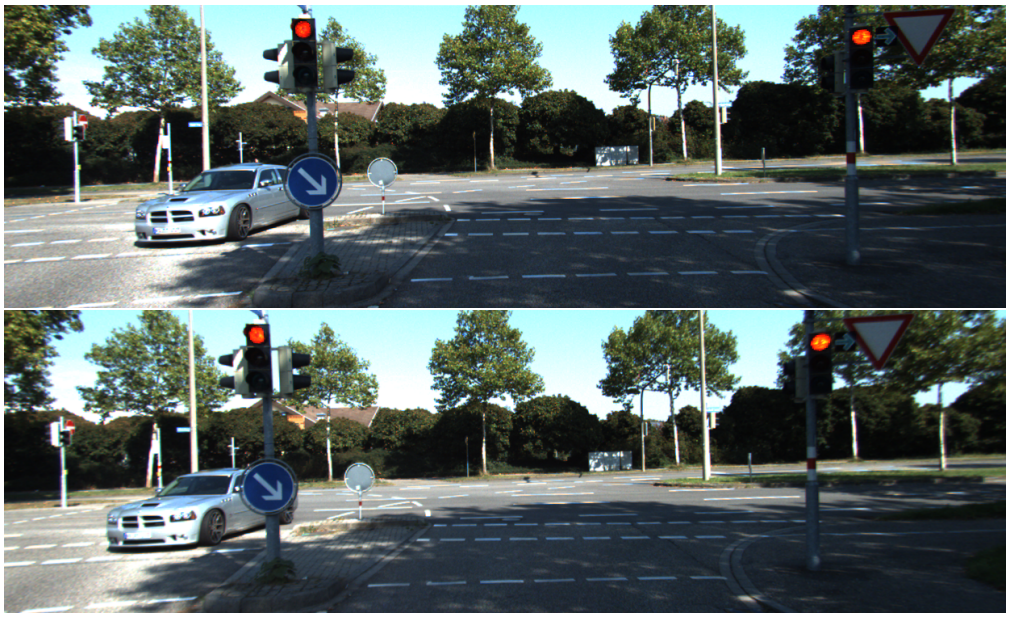

Figure 4: top: left image from the kitti dataset; bottom: right image

uses **opencv** function to compute the triangulation between the the projection of the 3D point into the two images:

```
P3D = triangulatePoints(Cleft,Cright,pleft,pright)
```

1. You have to compute 3D points from the opencv function (in the script `test_geom_kitti.py`)
2. Display the data returned by the opencv function and explain the line `Xcv /= Xcv[3]`.
3. Compute the distance between the two 3D points
4. Run the script (you have to select two points each image in the same order) and measure the diameter of the circular blue traffic sign.

5 additional question

Modify the previous script to select 4 points in each image and compute the area of the delimited shape.

Annexe : Transformation matrices

In the case of a homogeneous transformation, the mathematical model is a 4x4 matrix. Going from the reference frame R_i to the reference frame R_f is modelled by the matric ${}^iM_f$ by the relation:

$$ {}^iT_f = {}^iM_f = ({}^is_j \ {}^in_j \ {}^ia_j \ {}^iP_j) = \begin{pmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} {}^iR_f & {}^iP_f \\ 0 & 1 \end{pmatrix} \quad (1) $$

Using ${}^iM_f$, it is possible to compute the coordinates of a point P in the reference frame R_f, given its coordinates into R_f by the equation:

$$ \boxed{ \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}_{R_i} = {}^i M_f . \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{R_f} = \begin{pmatrix} {}^iR_f & {}^iP_f \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{R_f} } \quad (2) $$

When the transformation between the two reference frames is only a translation, the transformation matrix follows the notations:

- Trans(a, b, c) for a translation (a along the x axis, b along the y axis and c along z axis)
- Trans(x, a) for a translation of a along x axis
- Trans(y, b) for a translation of b along y axis
- Trans(z, c) for a translation of c along z axis

$$ M = \begin{pmatrix} & & & a \\ & I_3 & & b \\ & & & c \\ 0 & 0 & 0 & 1 \end{pmatrix} = \quad (3) $$

when the transformtion between the two reference frames is only a rotation, the notations are:

- $Rot(x, \theta_x)$ for a rotation (θ_x around the x axis)
- $Rot(y, \theta_y)$ for a rotation (θ_y around the y axis)
- $Rot(z, \theta_z)$ for a rotation (θ_z around the z axis)

In the homogeneous matrix, the rotation is modelled by the matrix R of the equation (1) 4. When there is no rotation, R belongs the identity matrix:

$$R = I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Example 1: A rotation θ_x along x axis.

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\theta_x & -s\theta_x \\ 0 & s\theta_x & c\theta_x \end{pmatrix} \text{ (autre notation)} \tag{5}$$

Let notice $(\underline{i_{R_f}}, \underline{j_{R_f}}, \underline{k_{R_f}})$ the basis associated to R_f and $(\underline{i_{R_i}}, \underline{j_{R_i}}, \underline{k_{R_i}})$ the basis associated to R_i. The rotation matrix R is given by the coordinates of $(\underline{i_{R_f}}, \underline{j_{R_f}}, \underline{k_{R_f}})$ in the reference frame given by $(\underline{i_{R_i}}, \underline{j_{R_i}}, \underline{k_{R_i}})$:

$$\begin{array}{rcl} \underline{i_{R_f}} & = & 1.\underline{i_{R_i}} + 0.\underline{j_{R_i}} + 0.\underline{k_{R_i}} = {}^i\underline{s}_f \\ \underline{j_{R_f}} & = & 0.\underline{i_{R_i}} + \cos\theta_x.\underline{j_{R_i}} + \sin\theta_x.\underline{k_{R_i}} = {}^i\underline{n}_f \\ \underline{k_{R_f}} & = & 0.\underline{i_{R_i}} - \sin\theta_x.\underline{j_{R_i}} + \cos\theta_x.\underline{k_{R_i}} = {}^i\underline{a}_f \end{array} \tag{6}$$

Example 2: A rotation θ_y along the y axis.

$$R = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix} = \begin{pmatrix} c\theta_y & 0 & s\theta_y \\ 0 & 1 & 0 \\ -s\theta_y & 0 & c\theta_y \end{pmatrix} \text{ (autre notation)} \tag{7}$$

Example 3: A rotation θ_z along z axis.

$$R = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c\theta_z & -s\theta_z & 0 \\ s\theta_z & c\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ (autre notation)} \tag{8}$$

Listing 1: HelloWord Python script: load an image and convert it using opencv then display using matplotlib

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Aug  6 08:46:29 2018
hello_wold simply loads an image with opencv, converts it to RGB and GRAYSCALE
and displays it using matplotlib
@author: thierrychateau
"""
import cv2
from matplotlib import pyplot as plt
import numpy as np
# This is an example of function with a string argument that returns another
# string
```

```
def my_func(my_arg):
    string2 = my_arg + ' toto ';
    return string2
# This is the main function that is launched when this file is run as a script
def main():
    # call function my_func
    st = my_func("my name is")
    print(st)
    # Read an image file (im is a numpy matrix
    # (nblines,nbcolums, BGR 3 channels))
    filename = "billard_large.jpg"
    img = cv2.imread(filename)
    # Convert to RGB
    imgrgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # Convert to grayscale image
    imggray=cv2.cvtColor( imgrgb, cv2.COLOR_RGB2GRAY )
    ## Display the RGB and gray images
    # divide plot screen into 1 line, 2 columns and select first subscreen as
    # current screen
    plt.subplot(121)
    plt.imshow(imgrgb)
    plt.title('Color RGB image')
    plt.subplot(122)
    plt.imshow(imggray, 'gray',vmin=0, vmax=255)
    plt.title('Grayscale image')
#
if __name__ == "__main__":
    # execute only if run as a script
    main()
```