Homework 05

Deepak Sharma
CSCI 731 Advance Computer Vision

August 1, 2017

Solution

1. **PartA** Camera Parameters:

   A. **Intrinsic Parameter**: Intrinsic parameters represents properties of a camera. In general these parameters doesn't change from image to image. It has been assumed that target camera is a euclidean camera, for which below four Intrinsic parameters has been identified.
   1. $\mathbf{c_x}$: X Focal length
   2. $\mathbf{c_y}$: Y Focal length
   3. $\mathbf{X_H}$: X coordinate of Image Center/Principle point.
   4. $\mathbf{Y_H}$: Y coordinate of Image Center/Principle point.

   Camera matrix is responsible for performing projection of 3D Camera coordinates points into image coordinates points and then translating into sensor coordinates points.

   Projection Matrix: Map camera coordinate(3D) points into Image coordinates points(2D).

$$ProjectionMatrix = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   Translation Matrix: Map Image coordinate points into Sensor coordinates points.

$$TranslationMatrix = \begin{bmatrix} 0 & 0 & X_H \\ 0 & 0 & Y_H \\ 0 & 0 & 1 \end{bmatrix}$$

   Camera Matrix: Combined Projection and Translation matrix.

$$\begin{bmatrix} c_x & 0 & X_H \\ 0 & c_y & Y_H \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4.49 & 0 & 2592 \\ 0 & 4.47 & 1728 \\ 0 & 0 & 1 \end{bmatrix}$$

B. **Extrinsic parameters**: Extrinsic parameters are responsible for converting World coordinate points into Camera Coordinates points. Hence they are image specific and show different value for each image. During the experiment 11 images has been used hence, 11x6 values have been shown in output file. For each image there are six parameters which perform a rigid body transformation on 3D coordinates. Three parameters for each translation and rotation.

Rotation Matrix: Perform rotation on World coordinate(3D) points while mapping into Camera coordinates points(3D). Rotation matrix has three degree of freedom in terms of **pitch**, **roll** and **yaw**.

$$RotationMatrix = \begin{bmatrix} R_{3X3} & 0_{3X1} \\ 0_{1X3} & 1 \end{bmatrix}$$

Translation Matrix: Perform translation on World coordinate(3D) points while mapping into Camera coordinates points(3D). Translation matrix has three degree of freedom in terms of **shift in x, y and z direction**.

$$TranslationMatrix = \begin{bmatrix} I_{3X3} & X_0 \\ 0_{1X3} & 1 \end{bmatrix}$$

Opencv camera calibration method reports six extrinsic parameters in a pair of tuples containing translation and rotation parameters for each image.

C. **Distortion Parameters**. In reality, a world coordinate points transformation into sensor coordinates can't be modeled by above described linear transformation due to the presence of non linear radial and tangential distortion, introduced into the image by the camera. Imperfection in camera lens produce redial distortion. Due to misalignment of lens and sensor axis images show tangential distortion. Since the nature of these distortion is non linear, they displace each point in image by a different amount of displacement. Displacement in a point from its ideal position has been modeled by bellow polynomial transformations.
Displacement due to Radial Distortion:

$$x_{distorted} = x_{ideal}(1 + k_1 r^2 + k_2^4 + k_3 r^6)$$

$$y_{distorted} = y_{ideal}(1 + k_1 r^2 + k_2^4 + k_3 r^6)$$

In the experiment, we have found value of below value of radial distortion parameters:
$\mathbf{k_1 = -2.11}$

$\mathbf{k_2} = \mathbf{5.39}$
$\mathbf{k_3} = \mathbf{-8.07}$

Displacement due to tangential distortion:

$$x_{distorted} = x_{ideal}(2p_1xy + p_2 + (r^2 + 2x^2))$$

$$y_{distorted} = y_{ideal}(p_1(r^2 + 2y^2) + 2(p_2^2xy))$$

In the experiment, we have assumed 0 tangential distortion for the given camera be setting **CALIB_ZERO_TANGENT_DIST** parameter in camera calibrate function of the Opencv hence values of tangential distortion parameters($\mathbf{p_1}$ and $\mathbf{p_2}$) were 0.

D. **Results:**


```
calibration_time: "Mon 24 Jul 2017 08:01:54 PM EDT"
nframes: 11
image_width: 5184
image_height: 3456
board_width: 8
board_height: 6
square_size: 1.
flags: 12
intrinsic_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 4.4911230010659183e+03, 0., 2592., 0., 4.4758131835436507e+03,
       1728., 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ -2.1062376520384640e-01, 5.3868313199758566e-01, 0., 0.,
       -8.0697629915430336e-01 ]
avg_reprojection_error: 1.1431361938667304e+00
per_view_reprojection_errors: !!opencv-matrix
   rows: 11
   cols: 1
   dt: f
   data: [ 6.12705410e-01, 4.87677634e-01, 2.83470464e+00,
       1.45126486e+00, 9.60515022e-01, 8.72121155e-01, 7.51025379e-01,
       6.86273634e-01, 6.10231578e-01, 5.32608449e-01, 4.95133460e-01 ]
extrinsic_parameters: !!opencv-matrix
```

```
rows: 11
cols: 6
dt: d
data: [ 6.0114871325493129e-03, 2.5751406267842534e-02,
    -9.0965342677986286e-03, -6.9825567103543007e-01,
    -6.9068017323364339e+00, 3.4552710050663592e+01,
    1.9327957070282772e-02, -1.1004194117907066e-02,
    -8.9697293411934963e-03, -6.7249902344137669e-01,
    -8.8326628042949444e+00, 5.0365702188069228e+01,
    1.7249961108606595e-03, -1.6707126722269067e-01,
    -3.3497817438206276e-02, -1.1437533684762466e+00,
    -4.6403053191216701e+00, 1.4239321307448881e+01,
    -1.2499269407024801e-02, -1.8765770557905723e-01,
    -3.6629512721594856e-02, -1.0758356554354978e+00,
    -5.0321510410340009e+00, 1.7738477980780779e+01,
    -1.0551159944545493e-02, -1.6151850903667569e-01,
    -3.3497792552555729e-02, -7.2948738708533589e-01,
    -5.6124947109611378e+00, 2.2974217792695555e+01,
    -1.0220316535930439e-02, -1.7792764545504478e-01,
    -3.4988275714528776e-02, -6.5669222758000956e-01,
    -6.0052193188140430e+00, 2.6353814532834633e+01,
    -9.4118019449910117e-03, -1.7143900673211765e-01,
    -3.3627294487496177e-02, -6.1535459325870656e-01,
    -6.4714798609307289e+00, 3.0483380714403506e+01,
    -8.0576433218868919e-03, -1.6994298949623624e-01,
    -3.2948138611811437e-02, -5.9444028284476635e-01,
    -6.9323013111465146e+00, 3.4372774911571078e+01,
    -3.3735180832278064e-03, -1.4094935565340952e-01,
    -2.9144574766354400e-02, -4.8280465752353852e-01,
    -7.3773438525378561e+00, 3.8041284127949311e+01,
    -3.6615479443345114e-04, -1.6024091325027437e-01,
    -3.0108241950246136e-02, -2.3319034455438056e-01,
    -7.7518400087028185e+00, 4.1502563903262221e+01,
    3.1920685756075037e-03, -1.9089383872835639e-01,
    -3.1861985095485684e-02, -4.3395249336508485e-01,
    -8.2006135041956956e+00, 4.5423975885075919e+01 ]
```

2. **PartB:** Using Above Camera parameters, Generated below Undistorted Image after Mapping.

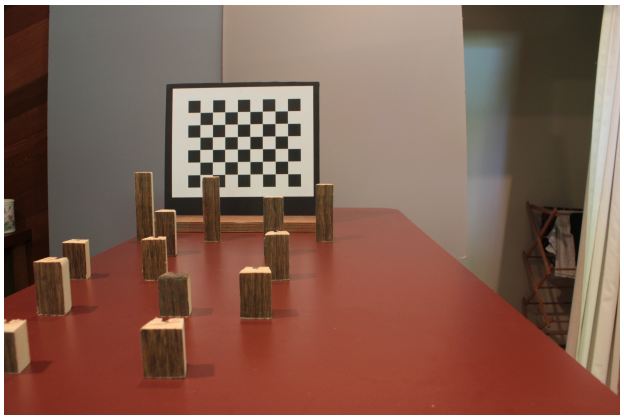

Figure 1: Input Image(Distorted Image)

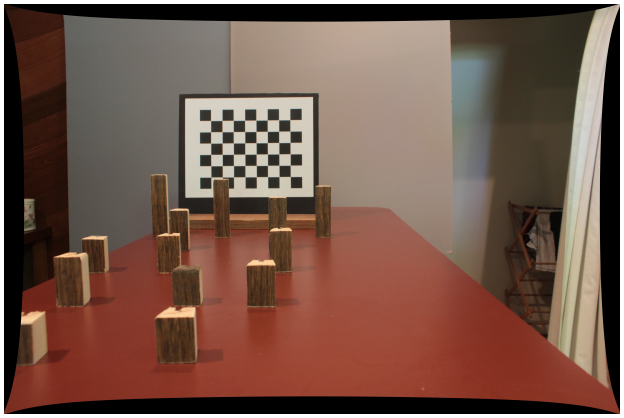

Figure 2: Mapped Image(Undistorted)

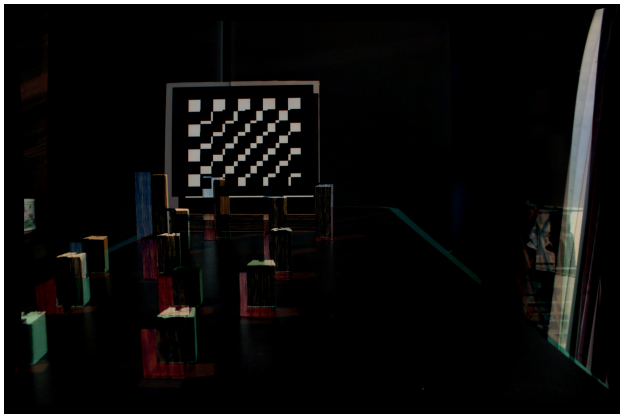

Figure 3: Difference Between Distorted and Undistorted Image

```
FileStorage fs( "camera_parameters.yml", FileStorage::READ);
   //fs.open("camera.yml", cv::FileStorage::READ);
   Mat cameraMatrixLoaded, distCoeffsLoaded ;
   Size imageSize;
   fs["intrinsic_matrix"] >> cameraMatrixLoaded;
   fs["distortion_coefficients"] >> distCoeffsLoaded;
   fs["image_width"] >> imageSize.width;
   fs["image_height"] >> imageSize.height;

   cout<<"Generating Maps"<<endl;
   Mat view, rview, map1, map2;
   view = imread(argv[1], 1);
   initUndistortRectifyMap(cameraMatrixLoaded, distCoeffsLoaded, Mat(),
                        getOptimalNewCameraMatrix(cameraMatrixLoaded,
                        distCoeffsLoaded, imageSize, 1, imageSize, 0),
                        imageSize, CV_16SC2, map1, map2);
   cout<<"Showing Results"<<endl;

   //undistort(view, rview, cameraMatrixLoaded,
   //       distCoeffsLoaded, cameraMatrixLoaded);
   remap(view, rview, map1, map2, INTER_LINEAR);
```

3. **PartC:**
Stereo Block Matching algorithm calculate a disparity map, this map defines the correspondence between the projected locations of a 3D world point into the left and right images. If a point is far from the cameras then the disparity between the locations of projection will be less in left and right images. If a point is near to the cameras then it will show high disparity. Using this phenomena we can measure the Z axis depth If we have access to a pair of images of the same scene, which are taken from two different lateral displaced points or arranging two different cameras into a lateral stereo geometry. Now triangulation method can be used for coveting a disparity map into stereo depth image.

$$Z(depth) = f.T(x_l - x_r)$$

Here: $\mathbf{T}$ is the lateral distance between the two cameras or the amount of lateral displacement provided to the camera before capturing the second image. $\mathbf{f}$ is focal length of camera/cameras. $\mathbf{x_l}$ and $\mathbf{x_r}$ are the location of projected point into the left and right image.
Block matching algorithm moves a sliding window along the horizontal epipolar lines for finding stereo correspondence between two points of the left and right images. Once it find correspondence it assign disparity for those points.

```
  bm->setPreFilterCap(31);
   bm->setBlockSize(61);
   bm->setMinDisparity(0);
```

```
bm->setNumDisparities(256);
bm->setTextureThreshold(10);
bm->setUniquenessRatio(15);
bm->setSpeckleWindowSize(100);
bm->setSpeckleRange(32);
bm->setDisp12MaxDiff(1);
```

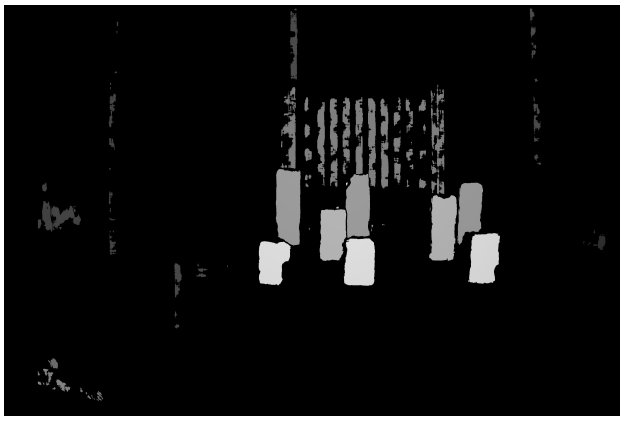

Figure 4: Disparity Map

4. **Source code credit**
   1. Camera Calibration [5, 1].
   2. Disparity Map [4, 1].
   3. Codding style and guidelines [2].
   4. Method definitions [1, 3]

References

[1] Gary Bradski Adrian Kaehler. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library 1st*. O'Reilly Media, Inc., 2 edition, 2016. ISBN 1491937998 978149193799.

[2] Errin Fulp. C++ coding guidelines, 2006. URL `http://csweb.cs.wfu.edu/~fulp/CSC112/codeStyle.html`.

[3] *The OpenCV Reference Manual*. Itseez, 3.2 edition.

[4] Andrey Kamaev. Demo code for disparity map, 2010. URL `https://github.com/npinto/opencv/blob/master/samples/cpp/stereo_match.cpp`.

[5] Steven Puttemans Adil Ibragimov taka-no me, Valery Tyumen. Demo code for camera calibration, 2016. URL `https://github.com/opencv/opencv/blob/master/samples/cpp/calibration.cpp`.