MEMORANDUM

To: Dr. Randy C. Hoover

From: Sharif Anani

Subject: Calibrating Stereo Cameras Using MATLAB

Date: OCTOBER 14, 2015

1 Summary of assignment

The original assignment was to use MATLAB's camera calibration tool in order to calculate the camera's intrinsic and extrinsic parameters. Since my partner and I's project involves stereo imaging, I decided to use the toolbox to calibrate both cameras of the stereo pair and calculate the transformation between the two cameras.

2 Steps taken to complete the assignment

2.1 Configuring the webcams

The first step was to configure the two webcams to work with MATLAB's image acquisition toolbox. This involves installing the webcam support package from the *Support Package Installer*.
The command `webcam` then returns a list of all available webcams, which we then pick out the webcams from.

2.2 Taking images using the stereo pair

We then need to take simultaneous images of the same scene using the stereo pair. to do this, both webcams were mounted side by side and the images were taking using a matlab script.

```
close all;clear all;clc
lst=webcamlist;
cam1=webcam(1);
```

```
cam2=webcam(3);
figure(1);
preview(cam1)
figure(2);
preview(cam2)
i=0;
numImagePairs=15;
imageFiles1=cell(15,1);
imageFiles2=cell(15,1);
for j=1:30
   fprintf('Capturing in %i seconds\n',30-j);
   pause(1)
end
while(1)
     pause(5);
     i=i+1;
     imageFiles1(i)={(snapshot(cam1))};
     imageFiles2(i)={(snapshot(cam2))};
     fprintf('captured %i images\n',i);
     if(i>=numImagePairs)
          break
     end
end
save('imdata.mat','imageFiles1','imageFiles2');
fprintf('Images saved to imdata.mat\n');
clear cam1
clear cam2
close all
```

The following two images are a sample from the images take from the two cameras.

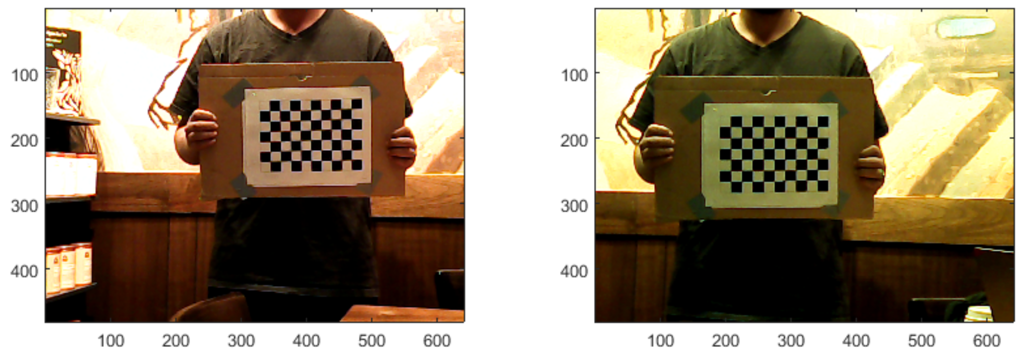


Figure 1: A pair of stereo images

2.3 Detecting the chessboard pattern

MATLAB 's toolbox provides a fast and easy way to detect the checkerboard on the images. The method is called `detectCheckerboardPoints`. This returns the image points of the corners and the checkerboad size. The script `getCheckers.m` shown below detects the points and shows them to the user for inspection in case the corners are erroneously detected.

```
close all
clear cll
clc
load('imdata.mat');
numImages=length(imageFiles1);
showCheckers=1;
% imPoints1=cell(numImages,1);
% imPoints2=cell(numImages,1);
for i=1:length(imageFiles1)
    images1(:,:,:,i)=imageFiles1{i};
    images2(:,:,:,i)=imageFiles2{i};

end


[imagePoints, boardSize] = detectCheckerboardPoints(images1, images2);

if(showCheckers)
    figure
    for i=1:length(imageFiles1)
        subplot(1,2,1),subimage(images1(:,:,:,i));
        hold on
        plot(imagePoints(:,1,i,1),imagePoints(:,2,i,1));
        subplot(1,2,2),subimage(images2(:,:,:,i));
        hold on
        plot(imagePoints(:,1,i,2),imagePoints(:,2,i,2));

        pause()
        hold off
    end

end
save('checkerData.mat','imagePoints','boardSize')
```

The following figure shows the chessboard detected on a pair of images.

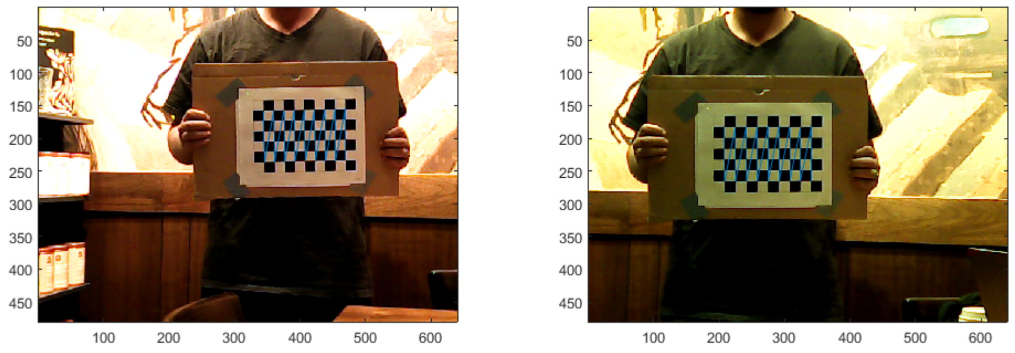


Figure 2: Checkerboard detected on a pair of images

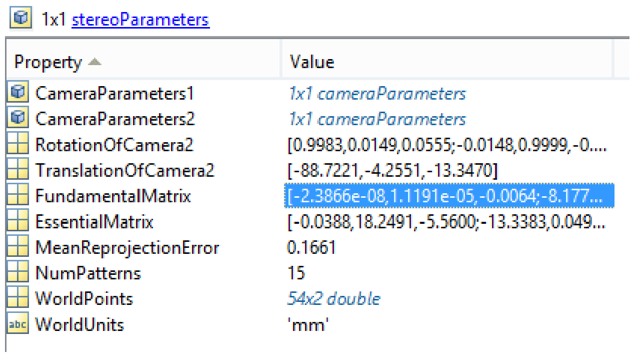
2.4 Obtaining parameters

This is the step where all the calibration is actually done.
MATLAB uses the function `estimateCameraParameters` to estimate the parameters using the image points (corners-pixels) and the world points - which are the points after applying a homogeneous transformation to estimate a 3d position for each corner. This correspondence is then used to calculate the rotation of the second camera w.r.t. the first camera.
The file `estimate3dCalib.m` below does that operation and saves the parameters.

```
close all
clear
clc
load('checkerData.mat')
load('imdata.mat')
squareSize=22.5;
% boardSize=[5,7];
% for i=1:length(imageFiles1)
%     images1(:,:,:,i)=imageFiles1{i};
%     points1(:,:,i)=imPoints1{i};
%     images2(:,:,:,i)=imageFiles2{i};
%     points2(:,:,i)=imPoints2{i};
% end

worldPoints=generateCheckerboardPoints(boardSize, squareSize);
stereoParams=estimateCameraParameters(imagePoints, worldPoints);
save('stereoData.mat','stereoParams');
```

The following figure shows the contents of the resulting calibrations.



1x1 stereoParameters

Property	Value
CameraParameters1	1x1 cameraParameters
CameraParameters2	1x1 cameraParameters
RotationOfCamera2	[0.9983,0.0149,0.0555;-0.0148,0.9999,-0....
TranslationOfCamera2	[-88.7221,-4.2551,-13.3470]
FundamentalMatrix	[-2.3866e-08,1.1191e-05,-0.0064;-8.177...
EssentialMatrix	[-0.0388,18.2491,-5.5600;-13.3383,0.049...
MeanReprojectionError	0.1661
NumPatterns	15
WorldPoints	54x2 double
WorldUnits	'mm'

Figure 3: Stereo calibration parameters

The first two members are the intrinsic parameters, extrinsic parameters, reprojection error, tangential distortion, rotation matrices, and other parameters for each of the cameras. The following figure shows the reprojection errors in the calibration.

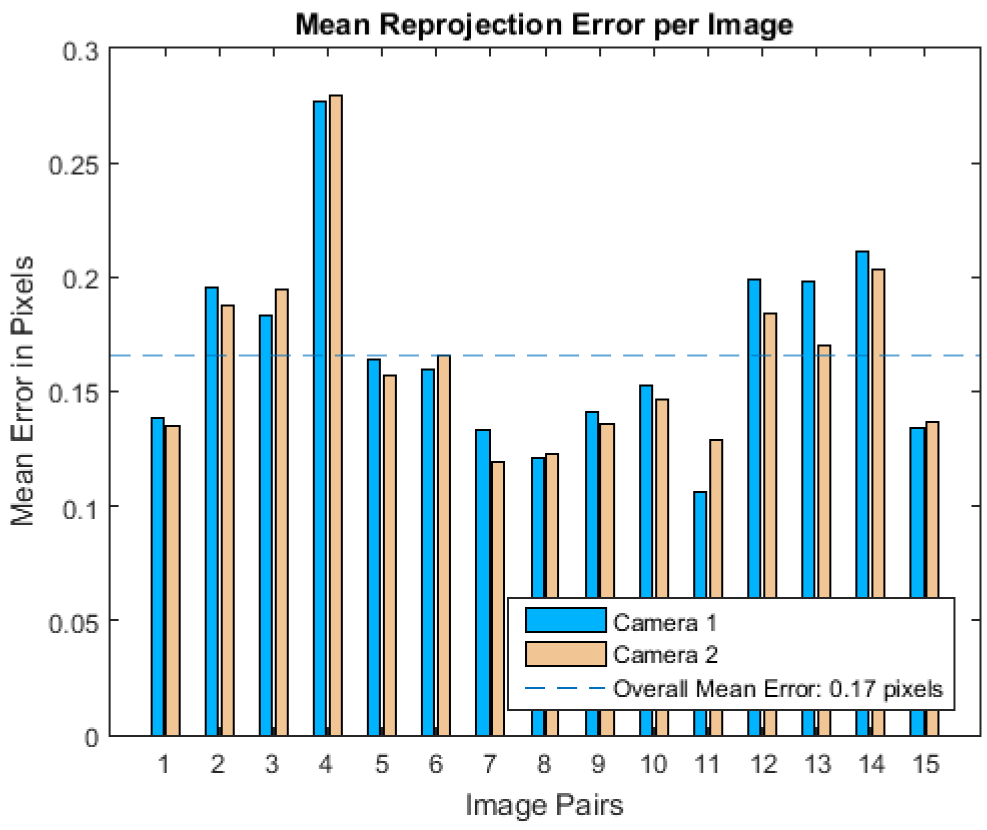


Figure 4: Reprojection Errors

3 Generating a disparity map

To generate a disparity map, the script `generateDisparity.m` was written. It takes two images, rectifies the, and generates a disparity map. The code used is included below:

```
close all
clear all
clc

load('imdata.mat');
load('stereoData.mat');

[J1,J2]=rectifyStereoImages(imageFiles1{1},imageFiles2{1},stereoParams)
    ;


disparityRange = [0, 64];
disparityMap = disparity(rgb2gray(J1), rgb2gray(J2), 'DisparityRange',
    ...
     disparityRange);
figure;
imshow(disparityMap, disparityRange, 'InitialMagnification', 50);
colormap('jet');
colorbar;
title('Disparity Map');
```

The following image is an example disparity map generated from the script.

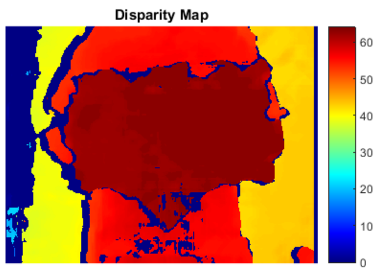


Figure 5: Example disparity map