AMME4701 Computer Vision and Image Processing
Tutorial Activity Week 5, 2018

THE UNIVERSITY OF SYDNEY

Camera Geometry and Stereo Vision

- This tutorial activity is to be completed during the Week 5 tutorial (Friday 31st August 2018)
- When you have completed the activity, you will need to show your results to the tutor who will check your work and check that you have sufficiently completed the task

Objectives

- This tutorial will introduce you to concepts in geometric camera calibration, two-view geometry and stereo vision.

1. Camera Calibration and Geometric Models

During the week 5 lecture we discussed geometric models for cameras, i.e. a set of equations that allows one to predict the expected pixel location of a point in an image given information about the point's location in 3D space, the camera's orientation and position in 3D space and parameters describing the camera's optical system. A common model that encompasses the primary focal properties of the camera lens, the distance between the imaging sensor, aperture and lens, and the distortion, and works for a wide variety of cameras is summarised in the week 5 slides. The model uses a pinhole assumption to represent the undistorted image coordinates (u, v) (slide 24):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} Z^c = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix}$$

followed by a parametric model for radial and tangential distortion to arrive at the real, distorted pixel coordinate in the image (u_d,v_d) (slide 34):

$$u_d = f_x x_d + u_0 \qquad v_d = f_y y_d + v_0$$

$$x_d = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2k_4 xy + k_5(r^2 + 2x^2)$$
$$y_d = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + k_4(r^2 + 2y^2) + 2k_5 xy$$

$$x = \frac{X_c}{Z_c} = \frac{u - u_0}{f_x} \qquad y = \frac{Y_c}{Z_c} = \frac{v - v_0}{f_y} \qquad r^2 = x^2 + y^2$$

Camera calibration is the process of determining the parameters associated with the camera's focal lengths (f_x, f_y), principle point (u_0, v_0) and lens distortion coefficients (k_1, k_2, k_3, k_4, k_5). During the tutorial, you will perform a calibration routine using a set of images captured using a stereo camera system.

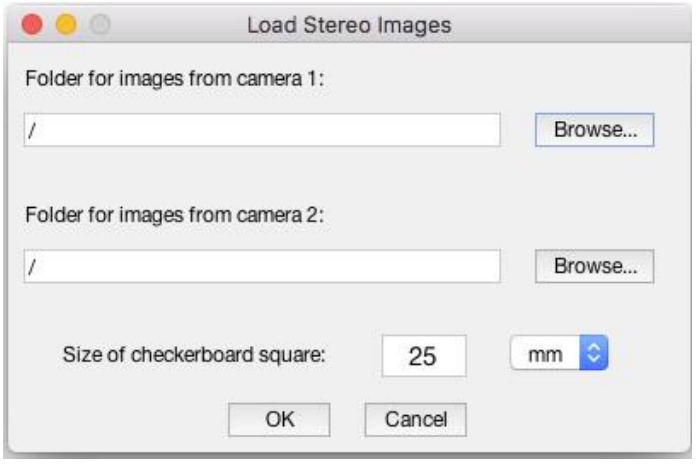
2. Stereo Vision: Calibration

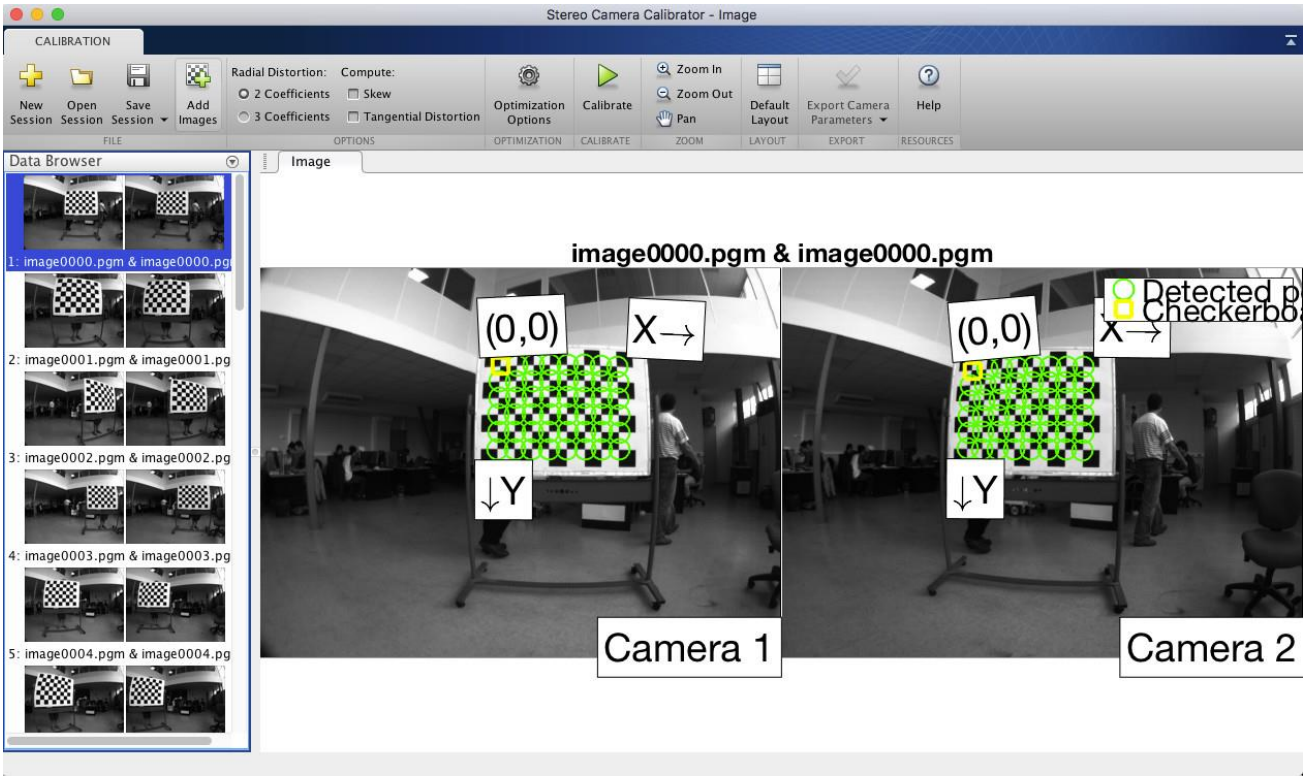
A stereo image system consists of two cameras that a rigidly-mounted to one another with a "baseline" distance between them, share a (roughly) common view and can be synchronously triggered. The calibration of a stereo-imaging system works in a similar fashion as for a single camera except that in addition to the single camera parameters, the extrinsic relative rotation matrix and translation vector (**R** and **t**) between the two cameras is also estimated.

MATLAB provides two apps for the calibration of cameras: the "Camera Calibrator" (for a single camera) and the "Stereo Camera Calibrator", an app designed for use with stereo camera systems. You can find the Apps by clicking on "Apps", "Image Processing and Computer Vision". Each app takes a series of checkerboard images captured from a variety of different perspectives, extracts the image coordinates of the corners between squares on the board, and uses these along with knowledge of the size of the squares to estimate the calibration parameters of the camera.

For stereo camera calibration, the calibration process relies on a set of image pairs (one image from the "left" camera and one from the "right" camera), each view containing the checkboard, and captured from a range of different perspectives.

Open up the Stereo Camera Calibrator app. After opening the app, you can select images from the stereo cameras by clicking on "Add Images". Separate folders are selected containing images from either camera 1 or camera 2 in the stereo pair, with matching pairs containing a common filename. The app also prompts the user to enter the size of the checkerboard square used.





The app then performs an automated procedure to extract the corner locations of each square in the checkboard, and associate these across camera 1/camera 2 images and across pairs. The app provides the ability to review the detected corner locations by clicking on the image pairs. By clicking the "Calibrate" button on towards the top of the app window, the intrinsic calibration parameters of each camera and the extrinsic parameters representing the relative rotation and translation between camera 1/camera 2 are then estimated using the extracted images coordinates of the checkerboard. Like the single camera calibration app, a 3D representation of the extrinsic parameters of each camera pair with respect to the checkerboard is provided along with a distribution of the residual errors in pixel locations per image.

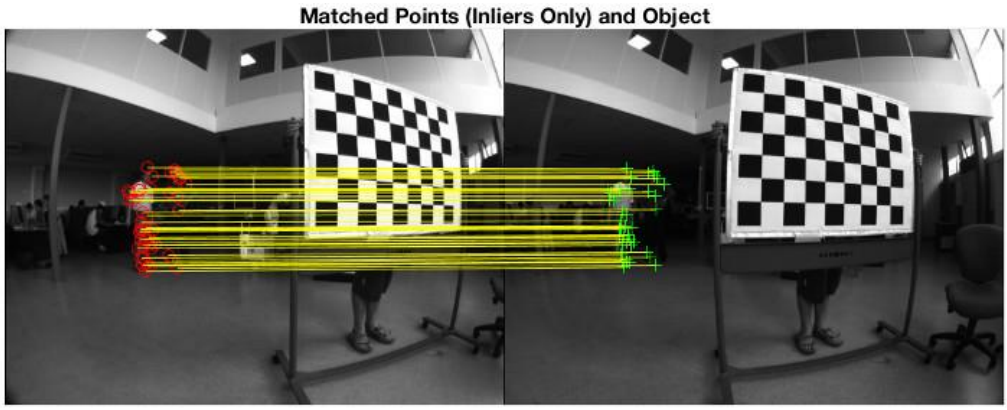
After calibration, the app allows for the user to export the calibration parameters as a variable to the MATLAB workspace. This variable contains fields associated with the intrinsic parameters (focal lengths, principle points and distortion parameters) of each camera and the rotation/translation of the second camera with respect to the first.

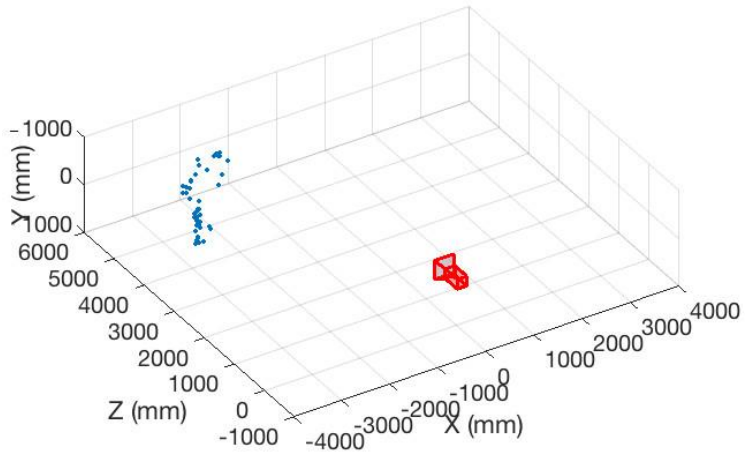
Tutorial Activity 1: Stereo Camera Calibration

The file "stereo_calibration_images.zip" contains images from a stereo camera, captured during a calibration routine of a stereo camera system attached to the Dala robot (courtesy of LAAS-CNRS). Unzip the file, and you should find two directories "left" and "right" containing images captured from a calibration checkboard.

- Open up the MATLAB Stereo Camera Calibrator. Load in the images from the left and right cameras and perform a calibration routine to determine the calibration parameters for the stereo system. Export these into the MATLAB workspace (as a variable `stereoParams`), and save the MATLAB variable to a .mat file. The size of each square on the checkerboard is 116 x 116 mm.
- Once you have the calibration parameters, try computing an undistorted image from one of the images in one of the stereo pairs. Use the function `undistortImage`, which takes as an input the original image array and the calibration parameters for the camera corresponding to the image (i.e. `stereoParams.CameraParameters1`). Compare the original distorted image to the undistorted image, noting the appearance of straight lines in the scene.
- Open up 'image0002.pgm'. Within this stereo pair, the world point p = [-4897, -35, 5411] mm (measured in the frame of reference attached to the first camera) corresponds to a certain object in the scene. Project this 3D point back into each of the two cameras in the stereo pair and plot the pixel coordinates of the point on top of both images. What physical object in the scene does this point correspond to?
- To find out, you should pass the 3D point through the projection equation incorporating the extrinsics and pinhole/lens distortion intrinsics for each camera. Start off by computing the undistorted pixel coordinate in each camera using the point position, camera focal lengths, principle point, and rotation/translation (for the second camera). Once this is computed, find the distorted pixel coordinates by using the distortion coefficients in the calibration parameters (have a look at the equations on slide 24/34). Although it is possible to perform this calculation using the function `worldToImage`, we want you to actually perform the calculations yourself in MATLAB (you can use this function to check your results)
- Note: MATLAB uses a slightly different matrix convention and ordering to the equations that have been presented in the lecture. If using the intrinsic matrix, you will need to take the transpose of this matrix and add the appropriate column of zeros to the end, in order to use it in the same matrix order as shown above.

3. Stereo Vision: Measuring 3D points from images





During the week 5 lecture, we explored algorithms for triangulating the 3D location of a point from two corresponding sets of pixel locations from a calibrated stereo system. We also saw how the epipolar geometry between two views could be used to apply constraints on the image coordinates of a point in one camera based on its image coordinate location in the first camera.

MATLAB provides implementations of a variety of stereo vision algorithms including triangulation and epipolar geometry-based outlier rejection. The function `triangulate` computes the 3D coordinate of an object from a matching set of pixel coordinates in two views and a set of stereo camera parameters. For example:

```
point3D = triangulate([u1,v1],[u2,v2],stereoParams);
```

Computes the 3D coordinate of a point seen at pixel location `[u1,v1]` in the first image and at pixel location `[u2,v2]` in the second image, where `stereoParams` is the variable containing the stereo camera parameters for the image pair.

To detect and match features (interest points) across stereo pairs for use in triangulation, MATLAB provides implementations of several commonly used interest point algorithms. The function `detectSURFFeatures` finds Speeded Up Robust Feature (SURF) points in an image, and the function `extractFeatures` returns descriptor vectors associated with these detected feature points. Once feature descriptors have been extracted, a set of matches between two sets of features in

two images can be computed using the function `matchFeatures`. For example, to compute a matched set of SURF features across two images:

```
points1 = detectSURFFeatures(im1);
points2 = detectSURFFeatures(im2);

[descriptors1, points1] = extractFeatures(im1, points1);
[descriptors2, points2] = extractFeatures(im2, points2);

matched_pairs = matchFeatures(descriptors1, descriptors2);
points1_matched = points1(matched_pairs(:, 1), :);
points2_matched = points2(matched_pairs(:, 2), :);
```

The function `extractFeatures` can also compute descriptors for a range of other interest point feature types that work in a similar fashion to SURF points.

The matches found using feature descriptors typically contain outliers (i.e. corresponding pairs of points that are not really of the same object in space). One way to detect and reject outliers is by keeping only a set of matching points that obey the epipolar geometry between the two image views. One way to enforce this is through a robust estimation of the fundamental matrix **F** representing the relative camera geometry using a RANSAC-style algorithm to estimate **F** from a set of point correspondences while finding the set of inlier correspondences.

The function `estimateFundamentalMatrix` provides the ability to compute **F** using a variety of different robust methods (including a RANSAC-like algorithm called M-estimator SAmple Consensus (MSAC)). Through the process of computing **F**, the function also returns a set of index values from the original set of correspondences that are inliers, based on their compatibility with the fundamental matrix. The function takes as an input variables containing the undistorted image pixel coordinates in each image, and returns the matrix **F** and inlier index values to the original sets of matched points.

Tutorial Activity 2: Stereo camera triangulation and depth estimation

From "stereo_calibration_images.zip" open up 'image0002.pgm'. We will explore the use of feature detection, matching, outlier rejection and triangulation methods to determine the 3D positions of the two people standing in the scene, just off to the left of the checkerboard.

- Extract SURF features from each of the camera 1 and camera 2 images. For the left camera, find features within a Region of Interest (ROI) that is isolated to an approximate box around the two people specified by (x,y,w,h) = [230,260,190,260] pixels by using the 'ROI' flag in the function `detectSURFFeatures`. For the right image, try specifying an ROI that takes up the left third of the image (i.e. (x,y,w,h) = [1,1,size(im,2)/3,size(im,1)]. Start off by extracting SURF feature points using the MATLAB function `detectSURFFeatures`.
- Use the MATLAB function `extractFeatures` to assign feature descriptors to each of the detected SURF points. Use the function `matchFeatures` to calculate corresponding pair matches based on these descriptors and plot these across the images using the function `showMatchedFeatures`
- Calculate a set of inlier correspondences based on this set of initial matches by using the function `estimateFundamentalMatrix`. Set the option 'Method' to 'MSAC' to use a sample consensus strategy called M-estimator Sample Consensus (MSAC), a variant on the RANSAC algorithm explored during the lectures. Note: you will need to undistort the points using the function `undistortPoints` before providing them to `estimateFundamentalMatrix`. Plot the sets of inlier correspondences using the function `showMatchedFeatures`
- Finally, compute the 3D locations corresponding to points located between the two images using the function `triangulate`. Use this set of functions to compute the average distance of the two people from the first camera in the stereo pair.
- Try going back and exploring the use of different feature types available (i.e. by using `detectHarrisFeatures`) and experimenting with the parameters for this feature type.

References and Further Reading:

[1] R. I. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2000

[2] D. A. Forsyth and J. Ponce, "Computer Vision - A Modern Approach", Prentice Hall, 2002 (Section 11, 12)

[3] R. Szeliski, "Computer Vision: Algorithms and Applications", Springer, 2010 (Section 7/11)

[4] MATLAB Image Processing Toolbox Documentation, https://au.mathworks.com/products/image.html, Mathworks, 2018.