Affine Adventure in Wonderland

"Take care of the sense, and the sounds will take care of themselves."
-- Lewis Carroll (*Alice in Wonderland*, Chapter 9)

Objective

Robotics is a fine and wonderful subject!

Its sensing need not be a mad tea-party. Indeed, it can leave one grinning like a Cheshire Cat. This problem set investigates sensor processing as well as the nature of observers in the context of **detecting**, **calibrating, localising**, and **tracking** regularly-shaped objects.

The problem set has two associated submissions and is to be completed **individually**. The aim of this problem set is to familiarise yourself with image processing techniques so that you are able to make a meaningful contribution to the team in the final project.

Tasks & Questions

Part A: Camera Calibration – Down the Rabbit Hole [40 marks]

In order to obtain a useful measurement, a camera needs to be calibrated. The calibration parameters that connect raw pixels images to 3D measurements are:

- focal length at the center ($\mathbf{f}_c$)
- principal point offsets from the center ($\mathbf{c}_c$)
- lens skew and distortion ($\boldsymbol{\alpha}_c$)
- Orientation ($\mathbf{R}_c$) and Position ($\mathbf{p}_c$) of the camera

Planar target camera calibration (Zhang's method) uses multiple viewpoints of a calibration target with calibration determined by correlating known points on the target with observed points by the camera. A common target (used by Matlab's camera calibration toolbox and OpenCV) is a black and white planar checkerboard as the corner points have high contrast allowing for (subpixel) precision even with noisy camera and target printing. Calibration will provide the intrinsics (perspective camera model parameters) and extrinsics (camera pose relative to the target). (See also, Lecture 7, Slide 25).

Task 1: Once Upon A Time…. (5 marks)

Use the provided 25mm checkerboard images to perform camera calibration on the sample images (and/or using a camera[1]). Return the camera intrinsics and extrinsics. You are welcome to use Matlab's Camera Calibration toolbox to find these parameters.

Task 2: A Look From Above (5 marks)

Using the information calculated above (Part A-Task 1), perform a transformation to generate a 'top-down' view of a scene image.

Task 3: Generalizing Calibration (10 marks)

Consider the calibration process (above) using a generalized *planar* checkerboard. The generalized checkerboard possesses **M** distinct features. The camera takes **K** images. You may assume an undistorted camera with following following characteristics:

- Zero skew (orthogonal pixel arrangement)
- Unity aspect ratio (square pixels)
- The image center is **unknown**.

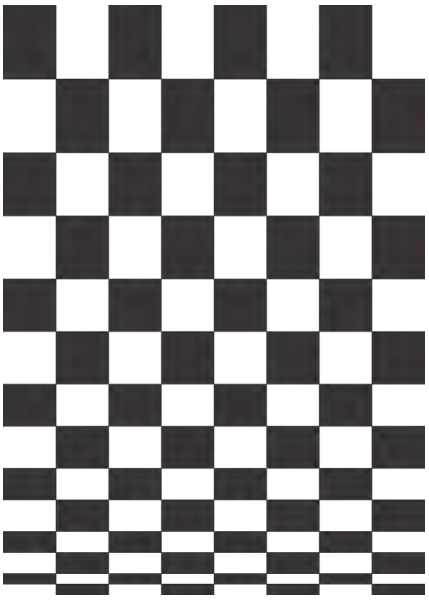
a. Given these assumptions, what unknown parameters are there to calibrate?
(Hint: what intrinsic parameters need to be recovered?
And, what extrinsic parameters need to be recovered for each image?)

b. For the **K** images, each of **M** features, how many constraints are given?

c. For the **K** images, each of **M** features, how many parameters need to be calibrated?

Task 4: "Calibrate Me" (20 marks)

Consider the "warped" calibration target shown at right [PDF Version]. We want to use this to calibrate the above camera (as described in Part A-Task 3)?



Are any modifications required to the standard Planar target camera calibration (Zhang's method) process (i.e. the method described in Part A-Task 1)? If so, what would this be?

In any case, please determine the minimum number of features (**M***) needed per image so as to solve a calibration and for that **M*** case, the number of images (**K***) need.

Then please develop a program that can calibrate a camera using this "warped" pattern. (Note: you are allowed to use or augment tools in Matlab, OpenCV, etc. to provide this.)

[1] Such as the UQ Robotics/Ximea Camera (in Lab) or one of your choice. Note, especially if you use your own, that the focal length should be held constant (i.e. manual mode) as it is one of the intrinsics being estimated. (**Extra credit question**: How could the process be changed to handle a changing focal length?)

Part B: Object Detection – Curiouser and Curiouser! [60 marks]

The goal of this part is to detect, count and estimate the pose (up to scale) of simple objects, such as a playing card(s). The task implicitly involves finding the dominant edges/features of the cards and then using this information to place a bounding box around them. Each type of card has a constant size (standard UQ EAIT card, of 56 x 87 mm), but its pose and appearance (colour) can vary. The cards of interest will be positioned "face up" (though other "face down" cards may be spread as clutter). For more skillful performance, we also seek to measure the distance (up to scale) between the cards.

Scene Structure

Basic performance shows standard understanding of the core concepts, whereas skillful performance is that which adeptly and automatically exploits dynamic structure. The workspace will be defined with varying levels of structure and clutter, with lower performance standards having more structure. This is outlined as follows:

	Basic Environment	**Skillful Environment**
Background	Generally uniform colour, similar to the basic environment test images	Colour may change between images and may be a different value than one in the test images
Clutter	Maybe (but sparse if present)	Assorted and spread throughout the scene. This must be ignored by your image processing.
Number of cards	1-3	0-6
Touching cards	No	Maybe
Environment	~2D	~3D

Basic scene example

Skillful scene example

Skillful Environment for (High) Distinction: You can select the environment that you wish to attempt - basic and/or skillful (see scene structure). Whereas operation at in a basic environment is expected of all submissions, performance additionally in skillful environments is needed to obtain (high) distinctions for Part B.

Task 1: Edge Extraction (5 marks)
For each image given, please extract the edges of the cards and items

Task 2: Bounding Box (20 marks)
For each image given, please determine an outline around the cards only. Marks will be deducted for errors including missed shapes, and, if testing in a skillful environment, for boxing clutter. For higher marks (high distinction) the bounding boxes should outline the card tightly (as compared to an image axis aligned rectangular box) and should also extend over occlusions.

Task 3: Pose Estimation (20 marks)
For each image given, please return the number of cards and the full pose of each one. For full marks, it should return location and orientation. Card location(s) should be given as the pixel coordinates of the center of the bounding/outline box. The orientation should be all 3 angles of the card relative to an image axis aligned camera frame located at the camera center. Note that for basic environments, which are essentially 2D, this may be just the card's planar rotation. However, for high distinctions in skillful environments, this should be all 3 angles.

Task 4: Playing it Straight on the Queen's Crooked Ground (15 marks)
This task considers the homography and the fundamental matrix. For each image given, please determine the **minimum** distance (up to scale, using a standard [L2] norm) between the cards as measured on the plane of the table. (Note: if there are less than two cards, then a zero answer is acceptable. Hint: Please see also §13.1-13.3 of MVG)

Extra-Credit Task 5: Who Stole the Fundamental Matrix? (10 marks)
Related to this, discuss "**Is the Fundamental Matrix pose invariant?**". That is, if both views/ cameras were rotated the same amount, then would the fundamental matrix between them stay the same before and after the rotation?
(Hint: Please see also §9.2-9.3 of MVG)

⇒ Programming Systems ⇐

You may elect to use programming languages and systems other than Matlab, such as C, Python, Julia, etc. (i.e. the class is language / system neutral). In particular, you may choose to use OpenCV. The default and supported programming system is MATLAB.

[FYI: The image at right is a coloured edgemap for fun ☺]

Submission & Due Date

For each Part, all code must be submitted in a single self-contained, independent zip file that can be run on a standard EAIT Workstation without any libraries/installations by the user. The code submission should include very clear documentation (a README file), installation scripts, etc. such that a tutor could easily open and execute your code. (Note: Tutors will not be tasked with debugging the script/environment. For example, if a python script solution requires installing external libraries (with `pip`), then clearly provide a list of fully specified commands for this).

In the README file, please **honestly indicate** which environment (i.e. Basic or Skillful) the submission is designed for. "Basic Environment Programs" that claim to work in skillful environments will receive lower scores than those that claim they can work only in Basic Environments, especially if they have no means of of working in "Skillful Environments".

Additionally, for each part a report must be submitted (in PDF format) that:

- Documents how the program works and shows **unmodified** results on some of the sample images;
- It should also briefly explain the solution and how it was found/computed (i.e., just printing the output alone is not sufficient).
- If it helps, you may include a flowchart or algorithm block that explains how your solution operates

Submit your code (**ZIP**) and Report (**PDF**) of your answers to the questions outlined in written submission via Platypus.

Final Submission: The problem set must be completed **individually** by
Friday, **September 28, 2018** (at 23:59 AEST).
Submission is via Platypus. Early submission is **highly** encouraged. ☺

Caveats

Some general "reasonable person" rules apply to the code and its execution:

- It is expected that you will use source/version control
- Internet access may or may not be present -- the code should assume that it will not have Internet access during execution and thus operate in a self-contained manner. This proviso excludes UQ license servers that may be needed by the program (e.g. Matlab). A "Mechanical Turk" or "phone home" solution is specifically disallowed.
- Codes with fixed (predetermined) estimates are not valid -- read "NO HARD CODING"
- Memory space may or may not be cleared between challenges and submissions -- The memory space might be cleared before each function. Thus, if your routines rely on parameters to be exchanged, it should do so by writing to a file. Similarly, if certain variables names (e.g. counters) are used between functions, then be sure to initialize them correctly.
- All source code(s) may be assessed -- Thus, it is requested that it is commented. If custom precompiled codes are used (e.g. mex files), the source code and compilation instructions should also be submitted.
- Computational and memory resources -- the functions should be able to operate reasonably on a "standard" Laptop/Workstation class computer (e.g. EAIT PC Workstations). Execution may be terminated after 5 minutes.