METR 4202 -- Advanced Controls & Robotics

Lab 2: Sensing & Perception --
Robotics: Spotty Perception

Vision is the Art of seeing Things invisible.
Jonathan Swift

Objective

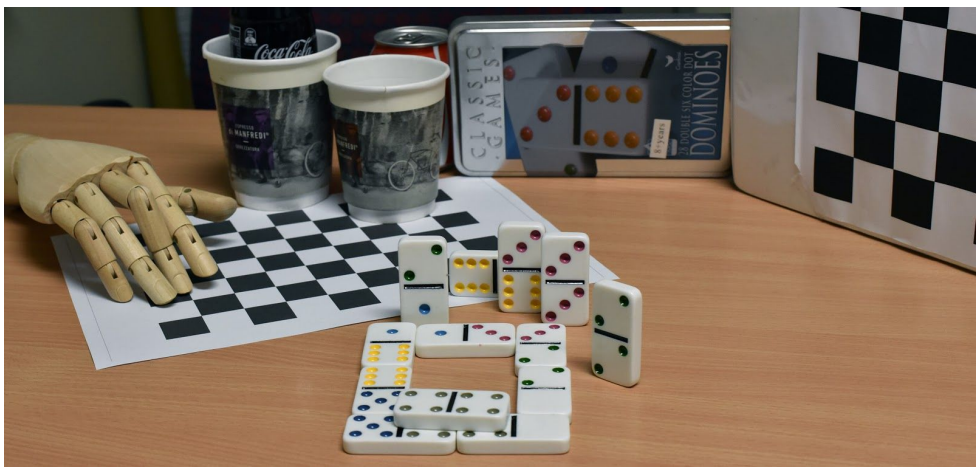
Sensing need not be left to chance (probabilistic methods notwithstanding). This laboratory investigates sensor interfacing and processing as well as the nature of observers in the context of **detecting**, **localising**, and **tracking** the a spotty object whose machinations may dominate the mind, Dominoes!

This is explored with the help of a RGB and/or RGB+D camera (or an equivalent sensor of the team's choosing). In this laboratory, teams will:

- **calibrate** the RGB or RGB+D camera
- **locate** the camera/sensor relative to a base frame
- **detect** the domino(s) in the scene
- **Locate** their position and pose, particularly in the presence of clutter and occlusions (i.e., "seeing things invisible")
- **track** the locations and rates of moving domino(s)

The laboratory has key programmatic sections – detection (i.e., "where are the Dominoes in the scene?") and object tracking ("what motions present?"). The laboratory may be done at different levels of performance ranging from basic to skillful (see scene structure). The former is required of all teams, whereas the latter allow teams to obtain (high) distinctions. Performance is seen as a cascade. Assessment will be done in stages (a "domino effect"): basic detection → basic tracking → skillful detection → skillful tracking → advanced operation(s).

See Domino. See Robot. See Spot Run. Run Spots Run! ☺

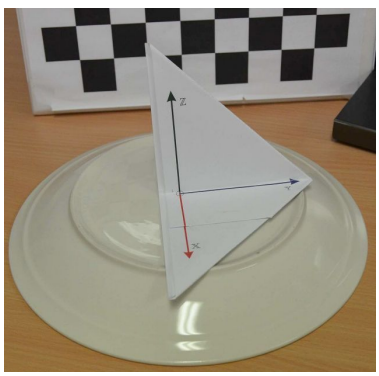


Scene Structure

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:

Item	Basic Level	Skillful Level
Background	Monocolor non-white paper (teams may remove it)	(Anything)
Clutter	No	Yes
Overlapping dominoes	No	Yes
Number of dominoes (Max)	3	7 (except challenges)
Colour calibration target	Optional	No (except for core task 1)
Calibration pattern	Provided	No
Central frame placement	Static (Optionally team-placed)	Dynamic (Random)
Domino poses	2D	3D (arbitrary inclinations)
Domino stack depth (Max)	1	2
Domino speed (Max)	5 cm/second (0.05 m/s)	30 cm/second (0.3 m/s)
Lighting variations	Room	Room + Flash/Spotlight(s)

Central Frame



The scene will consist of a central frame. By default, it is defined as being right handed with the *x-axis* defined as being oriented towards the camera (parallel to the table surface) and the the *z-axis* upwards (presumably in line with the gravity normal) and the *y-axis* being orthogonal to these axes (see illustration at left). It may be set by the team except in the most advanced level (master) of operation.

The central frame might not be on the table surface. Its location may be static (meaning that it doesn't move) or dynamic (meaning that the tutors may move the origin between attempts).

Calibration

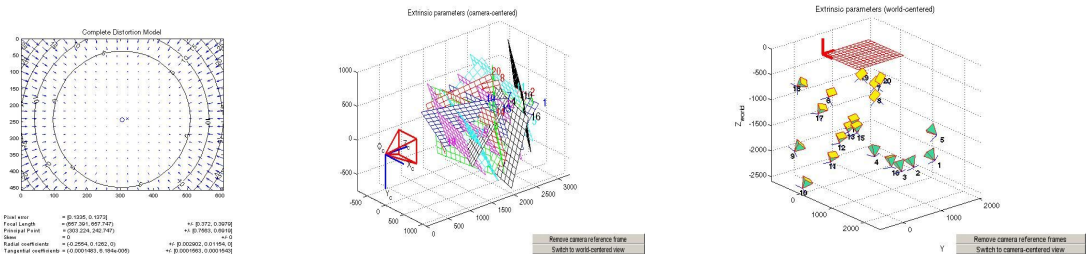
In order to obtain a useful measurement, the Kinect needs to be calibrated first. Let us begin with the camera. As noted in class, the calibration parameters to connect raw pixels images to 3D measurements are:

- focal length at the center ($\mathbf{f}_c$)
- principal point offsets from the center ($\mathbf{c}_c$)
- lens skew and distortion ($\mathbf{\alpha}_c$)
- 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 the 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. In this laboratory, while camera placement is an option, the view is fixed. Thus, the camera has to be calibrated before; or, the target needs to have motion (e.g., be attached to a soft spring, such as a string holding the target, or even via a mobile); or, there has to be be multiple viewpoints (or calibration targets) in the scene (e.g., a calibration cube).

Calibration will provide the intrinsic (perspective camera model parameters) and extrinsics (camera pose relative to the target). However, one might ask if this is necessary, as one way to frame the problem is to put the camera well overhead, thus making the scene essentially para-perspective. While this could even make calibration a direct linear (scale) operation, it comes at the cost of less occlusion robustness and no direct means for radial distortion correction. An example calibration (from the Bouguet Calibration Toolbox) is shown below:

```
Calibration results after optimization (with uncertainties):
Focal Length:          fc = [ 657.39071   657.74678 ] ± [ 0.37195   0.39793 ]
Principal point:       cc = [ 303.22367   242.74729 ] ± [ 0.75632   0.69189 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]  …
Distortion:            kc = [ -0.25541   0.12617   -0.00015   0.00006  0.00000 ] ±
  [ 0.00290   0.01154   0.00016   0.00015  0.00000 ]
Pixel error:          err = [ 0.13355   0.13727 ]
```



The basic level of calibration would be calibrate this in advance before the laboratory starts, whereas the more advanced level would be to calibrate camera online during the laboratory.

Core Tasks: Of Tiles and Suits

The core tasks in this laboratory are focused around basic concepts of the geometry of perception. (the "sense" of "sense-control-act"). To borrow the vernacular of dominoes, performance is measured "tiles". Core objectives:

(1) **Calibration**: camera geometry (intrinsics calibration)
(2) **Sensor Placement**: Where to place the sensor? And, a recovery of sensor **location** and **pose** (extrinsics calibration)
(3) **Object Detection**: Determining if there domino(es) in the scene, particularly in the presence of noise and clutter, and the their values
(4) **Object Mapping**: Determining the (ideally metric) position(s) of domino(es) in the scene
(5) **Motion Tracking**: Reporting the (ideally metric) motion (i.e., speed) of domino(es); at advanced levels in the presence of an occluder (i.e., the correspondence problem).

Notes:

- The number of "tiles" next to each level is indicative, "up to" and **not** absolute.
- In general, basic levels correspond to basic level environments, whereas Master level corresponds to advanced levels of performance.
- Except for the eigen level, the frame may not be placed on the object of interest.

Details:

1. Camera Calibration

Teams should **fully** calibrate their camera. Teams are allowed to choose their pattern, such as an A4 **30 mm** square checkerboard pattern.

- **Basic Level** (1-2 tiles): Calibrate the camera in a "Basic Level" environment. The teams may move the target themselves. While automatic operation is recommended, semi-manual operation (1 tile) is allowed. Automatic calibration (2 tiles) of the camera may allow manual pattern motion (i.e., "calibration dancing").
- **Advanced Level** (3-4 tiles): Fully automatic calibration of the camera in an "Advanced Level" environment. Manual pattern motion is also allowed. For four tiles, this should be done in a manner that reports the number of views needed and current level of accuracy to the user such that process may be optimal with regards to calibration time (views).
  Thought Experiment: What is the minimum number of views and the minimum number of points that need to be tracked in these views necessary for calibration?

2. Sensor Placement

The system should locate the camera (in metric coordinates) relative to the central frame. Importantly, teams must also be able to **quantitatively** justify sensor placement. Teams may provide a central frame target of their own choosing (i.e., using any given design or their own as long as the frame chosen does not obstruct operation/marking). Teams may choose to place the frame manually, but scores will be capped at 1 tile.

- **Eignen Level** (0 tiles): The central frame is placed at the camera origin.
- **Basic Level** (1 tiles): The central frame is placed in the workspace and if a target is used, then the target is visible by the camera. The estimated location is within 10 cm (total straight line distance) of the actual value (as surveyed by the tutors)
- **Advanced Level** (3-4 tiles): The central frame is placed by the tutors. The estimated location is within 10 cm of the actual value. Pose estimates are attempted. For four tiles, the system also attempt determines pose. The estimates are within 5 cm and ±15°.

3. Object Detection
The system should be able to detect if there is are domino(es) in the scene. If there are domino(es) present, the system should then attempt to recover their value (i.e., identify which one the 28 dominos in the set it is).

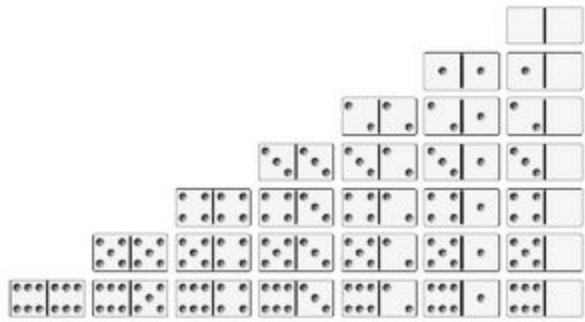


Figure 1: Standard Double-Six Domino Set (28) [image source]
(Note: Domino Type is also part of Unicode)

- **Basic Level** (1-2 tiles): The system is able to detect that a single domino (randomly selected by tutor) is in the scene or not. Operation is in a basic environment. Dominos are flat on the table, separated and not stacked. For two tiles, it should report the value of the domino.
- **Advanced Level** (3 tiles): The system should detect the presence of multiple dominoes at any arbitrary pose. Operation is in a basic environment. For full credit, it should return the value of the tiles.
- **Master Level** (4-5 tiles): Now in an Advanced environment, the system is able to automatically detect the presence of a domino in any arbitrary pose. It should be able to detect dominoes in the presence of clutter. For five tiles, it should do this robustly and quickly with little, preferably no, error.

4. Object Mapping
The system should be able to return an estimate of the (ideally metric) location between dominoes
- **Basic Level** (1-2 tiles): The system is able to return a 2D estimate of the domino's location in pixels (1 tile). For two tiles, this is relative to an arbitrarily placed base frame and/or in metric units (not pixels). Operation is in a basic environment.
- **Advanced Level** (2-3 tiles): The system should return the 3D metric position of the multiple dominoes relative to the base frame (from Core Task 2). For three tiles, operation should be in an Advanced Level environment. This is without obvious occlusions.
- **Master Level** (4-5 tiles): This must be an Advanced environment, the system is able to return the automatically the 3D metric pose (i.e., position and orientation) of dominoes even in the optional presence of occlusions (or stacking).

5. Motion Tracking
The system should give an estimate of the motion of domino(es).
- **Basic Level** (1-2 tiles): The system is able to return a 2D estimate of the pixel "speed" of a single moving domino in a basic environment. For two tiles, the system track multiple dominoes (in pixels) and/or return an inter-domino value in metric units.
- **Advanced Level** (3-4 tiles): The system should return the metric speeds of multiple dominoes. For four tiles, this should be in an Advanced Level environment.
- **Master Level** (4-5 tiles): This must be an Advanced environment, the system is able to return the automatically the 3D metric speed of the dominoes even in the optional presence of occlusions (or stacking).

Challenges: Lifting Our Game!

It is expected that the challenges are done in an advanced environment (up to 2 tiles), but partial credit will be given in basic environments (½ tile). A reasonable attempt is one where the solution has a likelihood of producing a correct result. Whereas robustly successful solution is one that yields correct results reliably under varying conditions.

(1) **Of Dice and Men**
- The system should be able to differentiate between dominoes and dice.

(2) **No Bones About It!**
- The system should be able to differentiate between a picture of domino(es) and the actual physical dominoes.

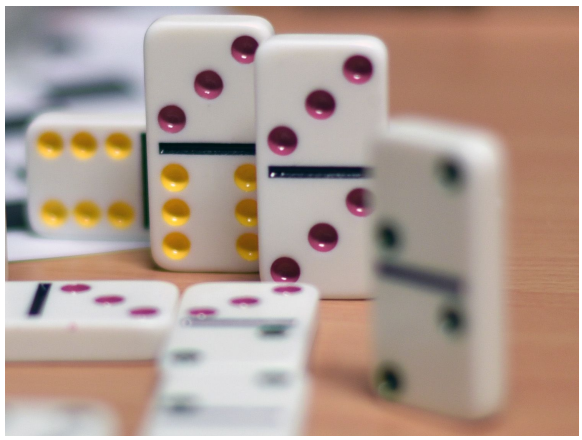
(3) **Happy Dominos**
- The system should operate in the presence of a varying, remote-controlled Multi color LED illumination source.

(4) **Shadowboxing**
- The system should track shadows and place a box around them.

(5) **Mobile Domination**
- Construct a mobile such that it has at least three dominoes in motion plus at least two non-domino objects are in motion. The motion of dominoes must be estimated. It may be, but need not be, red.

(6) **Quality over Quantity**
- The system is able to tell if a set of 28 dominoes, say spilled from an arbitrary box, is a proper and complete set (see also Fig. 1, one may assume they are face-up and not stacked, but they may be in arbitrary order).

(7) **Domino Effect**
- The system is able to count (and ideally recognize) dominoes standing in a domino effect train. The number of dominos may range from 2 to 28.



Marking: Assessment Criteria for Overall Lab Mark

Despite the hand-wringing, let us consider the process of handing over grades:

In short for the following grade levels:

- 1-3: Teams attempt and are somewhat successful at 1 of the "core tasks".
- 4: Teams attempt some and are marginally successful at least 2 "core tasks"
- 5: Teams attempt most "core tasks" and are successful at least 2 "core tasks"

- 6: Teams attempt most "core tasks" in an advanced environment and are robustly successful at most (at least 3) core tasks even, including the presence of noise/clutter/etc. They attempt at least one Challenge Task.
- 7: Teams robustly and successfully complete all "core tasks" in an advanced environment and attempt at least 2 Challenge Tasks such that at least one completed successfully in an advanced environment.

As a rough guide that mapping between tiles and the grades is:

Grade	Advanced Level	Description
2 (20-45)	2 tiles	At least one task performed. For example, you are able to calibrate the Kinect in both color and metric space.
3 (45-50)	3-6 tiles	Very substandard performance, For example, you are able to detect the presence of objects in one basic scene.
4 (50-65)	7-10 tiles	**Basic level operation.** For example, you are able to detect the location of at least one target object in some of the scenes with satisfactory accuracy.
5 (65-75)	11-15+ tiles	**Intermediate operation level.** For example, you are able to detect the location of a Domino in the scene with good accuracy.
6 (75-85)	12+ tiles + ≥1 Successful Challenge	**Very good intermediate to Advanced Level performance.** For example, you are able to detect the location of some target objects in each scene with great accuracy.
7 (85-100+)	15+ tiles + ≥2 Successful Challenges	**Excellent performance**. Most of the tasks are attempted well. Teams are able to robustly detect the locations of objects in complex scenes with superb accuracy.

Workspace

The workspace is similar to Lab 1 with extended height. The base plane is 320mm × 192mm (a standard Lego Mat area). The height is 500 mm (or about five times higher). Especially for operation at more advanced levels, it may include clutter in the form of lego pieces, marbles, cups, coins, soda beverage cans, small chocolates, and other small random items that come to play (and in homage of previous years' METR4202 course projects).

Teams and Groups

The project will be conducted in **teams** (up to six maximum). Notionally these will be based around Laboratory 1, but you may also choose an individual from another group (or laboratory session) as long as you understand that you may not be able to work together for the final project that will draw upon work completed in this one. A requirement to pass this Laboratory is that you must be in a team and that team must be registered in Platypus[2] by September 12, 2016.

Other Programming Systems and Cameras

Teams may elect to use programming languages and systems other than Matlab, such as Visual C or Python (i.e., the class is language / system neutral). In particular, teams may choose to use OpenCV.

Teams may choose to operate the Kinect's RGB camera in high resolution mode (i.e., they may use the 1280x1024 mode also provided by the MS Kinect SDK as compared to the 640x480 default mode provided by the OpenNI SDK). Similarly, teams may use another (web)camera on the proviso that the camera is autonomous (i.e., it can take pictures without manual intervention) and that the maximum resolution is set to (or automatically down-sampled to) 1280x1024. (n.b., the allure of high-resolution can be a trap in vision and signal processing applications as this comes with higher data processing requirements and often comes with more noise).

In both cases, however, the only supported programming system and hardware are MATLAB the the Kinect camera (using OpenNI/Primesense drivers).

External Sites/Programs

Some external programs and site that might help with the process are:

- **CLAMS -- Calibrating, localizing, and mapping, simultaneously**
- **Sample videos of an advanced scene taken with different camera types** (as zip)

Standing on the Shoulder of Giants

Domino recognition and robotics has been explored in the domain. You are encouraged to adopt a principled approach and use cited-citations. Please be sure to reference appropriately.

- Stanford Domino Playing Robot (DomiBot): News Article | Video | Github repository

Sample Videos

Some sample videos, taken with different types of video cameras of a very advanced scene taken with different camera types is available as ~1.4 GB zip file (from here). Thank you! :-)

Demonstration

As with laboratory one, the system will need to be demonstrated. As with Laboratory 1, there will be signup times on Monday, October 3 and Thursday, October 6 (i.e., during teaching week 10). During the demonstration period, teams may choose to demonstrate the focus area tasks in **any order** they choose. For each of the tasks, teams may repeat a task once if they choose; however, the team receives the value from either not both (i.e., repeat task demonstrations do not add).

Given the number of teams, the demonstration times (of 20 minutes total including setup, leaving 10 minutes for discussion) will be strictly enforced. It is recommended that teams come 30-45 minutes in advance of their demonstration appointment. It is also recommended that teams practise their demonstrations as time limits will be enforced even if teams have not been able to demonstrate their solutions to the tasks (i.e., teams will receive grades on the solutions they demonstrate; not the solutions they might have had, but did not deliver).

Individual Grade

Each member's individual grade will be computed from: The average assessment of the team by the teaching staff at the demonstration, The peer assessment factor, and the report.

The final score is then computed as follows: Individual Normalized Grade =
([PAF] * [Lab Demonstration (out of 100)] + Group Lab Report (out of 20)) * …
([Total Score to Normalize to a Percentage (in this case 1/125)])
[PAF] = (([Number of Team Members] * [Individual PAF]) / ([Sum of PAFs for the Group])) ^ 0.60

Due Date

The lab has been **extended by one week** so that it should completed and demonstrated notionally in the Week 10 Practical/Lab session (i.e., **Thursday, October 6, 2016 or Friday, October 7, 2016** (for those typically in the Monday session)). There will be a sign-up sheet for demonstration times. A short, brief **individual** report (~1 page, details in class) will also be submitted. It will be due by 11:59 pm on **Saturday, October 8, 2016** via the Platypus submission system. Early submission is **highly** encouraged.

Support Link

During the lab sessions, please visit https://goo.gl/dpSqN5 to queue questions. Thank you! :-)

Deliverables & Submissions

It is also required that all code be submitted (via version control system, such as Git). As the laboratory is language and camera neutral, teams that do not use Matlab should provide very clear documentation and README files with their submissions.

Output has to be in a human readable format. An **example** structure might be:

1. **Camera Calibration:** given a sequence of RGB+D data containing checkerboard patterns from an uncalibrated Kinect (or RGB for a camera), return the intrinsic and extrinsic constants of the Kinect. The returned intrinsic properties should be in a structure like following, the square brackets indicate the expected size of each property. Note that the extrinsics has three dimensions; the third dimension has a transformation matrix for each RGB calibration input image:
   intrinsics.fc = [1x2], intrinsics.cc = [1x2], intrinsics.alpha_c = [1x1],
   intrinsics.kc = [1x5], intrinsics.err = [1x2], extrinsics.transformation_matrices = [4x4xN]

2. **Detection:** given an image from the Kinect/camera, it must return the Domino location(s) in a standard format.

3. **Localisation and Mapping**: given an image from the Kinect/camera, it must return the location of the camera relative and/or domino(es) (depending on the mode) relative to a central coordinate frame in metric coordinates (mm) with orientations in (degrees). There should be a row for each domino located. There should be 3-6 columns to represent the domino's pose: [*xPosition, yPosition, zPosition, Roll, Pitch, Yaw*]. Orientation may be in Quaternions, Euler Angles, Rotation matrices, etc. if specified in the header or function documentation.

Judges

The course coordinator, lecturers and tutors will act as judges. The course coordinator will act as chief judge. All decisions made by the judges will be final, and no correspondence will be entered into. Contestants may approach the organiser about possible designs that may be questionable under the rules listed above. Any queries will be treated with the utmost confidentiality and will not be divulged.

Custom Levels

As custom level ideas are sent in and approved, they will be posted here for the benefit of other teams.

- Open Source Code -- The entire code base is properly documented and shared on a public, open-source repository (e.g., GitHub) = +1 tile

Caveats

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

- It is expected that teams will use source/version control
- Codes with fixed (predetermined) estimates are not valid (even if the value is correct).
- The use of the Matlab Camera Calibrator App may be used. However, teams will still be responsible for being able to explain how the calibration process works, particularly intrinsics, extrinsics, the minimum number of frames and points required (see also § 2.1.5 (pp. 45-49) of Szeliski, *Computer Vision: Algorithms and Applications*, 2010).
- 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 explicitly disallowed.
- 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.
- Each team's submitted functions will be run in their own directory -- Reading other teams' files or memory is disallowed.
- 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 (e.g., makefiles) should also be submitted.
- Computational and memory resources -- the functions should be able to operate reasonably on a "standard" Laptop/Workstation class computer (such as the UQ EAIT PC Workstations). Judges may terminate execution after 5 minutes.

METR 4202: A Robotics Domino Effect!