Fair Dice: A Tilt and Motion-Aware Cube with a Conscience

Kristof Van Laerhoven and Hans-Werner Gellersen
Lancaster University
{kristof, hwg}@comp.lancs.ac.uk

Abstract

As an example of digital augmentation of a tiny object, a small cube-sized die is presented that perceives and records what face it rolls on. It is thus able to detect bias and compensate for unfair behaviour due to its physical imperfections. On a deeper level, this case study demonstrates the integration of energy-efficient sensor fusion, and a wireless interface to adaptive classification heuristics.

1. Introduction

"*The chance element in thousands of indoor games is introduced by a variety of simple random-number generators. The most popular of such devices, ever since the time of ancient Egypt, have been cubical dice.*" (Martin Gardner, Mathematical Magic Show).

Which begs the question: How fair are these dice? Does each face appear with equal probability when it is tossed? And how can one find out whether particular dice are fair or not?

Dice are thrown to provide uniformly distributed random numbers. Fair dice need to be symmetrically shaped and have a centre of gravity in the exact middle. The methodology that dice manufacturers apply to achieve fair dice, is based on making near-perfect smooth and symmetrical cubes of a homogeneous material. However, even with these measures in place, no single die is perfect.

2. Unfair dice

Intentionally unfair dice are called "loaded" or "crooked" dice: they are altered to produce skewed or predictable results, for cheating or entertainment. Some have round and sharp edges and slightly-off square faces, others have weights added to one side.

More sophisticated versions of the latter type include "tappers", which have a drop of mercury load in the center of the cube that is activated by tapping the die, and variable loaded dice, which are hollow with a small weight and a wax-like weight, allowing the cheater to change the die's centre of gravity by breathing on it or holding it in hand. Inserting a magnet into the die and embedding a coil of wire in the game table is a more modern version of a variable die.

Dice that are produced using a substandard material that wears down over the years, or that have been polished insufficiently or cheaply, also risk being or becoming unfair.

Even if the dice themselves are very close to fair, environmental features and circumstances might still make the dice throw unfair: Composer Jeremiah Clarke, for instance, mentioned in his suicide note that his coin flip to pick the suicide method landed edge first in the mud (which he took as gunpowder) [2].

3. Fair dice

This paper proposes to tackle the problem of unfair dice by giving them a "conscience": they are digitally augmented to: (1) detect throws, (2) recognise on which side they landed, and (3) keep a history of their previous throws. After an appropriate number of samples are taken, our augmented die can alter its outcome to produce an unbiased result. This paper's

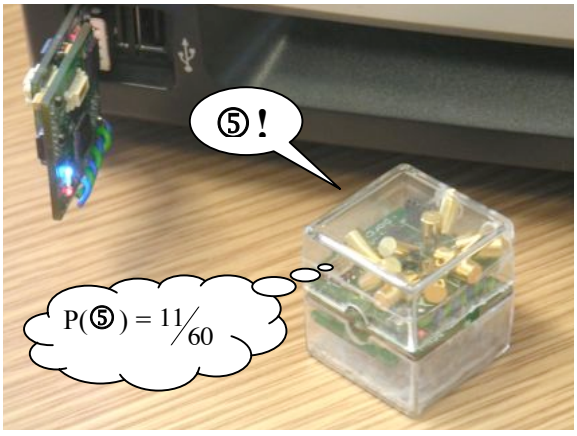

Figure 1. A cube does all sensing and processing internally, and then reports the results wirelessly.

die transmits its result via a wireless link to a screen. In a more advanced version, this could mean that the cube could adjust an internal weight just like the variable loaded dice, so that no external components are required to display results.

Before covering the application of an augmented die, we start with how ideal dice should behave.

4. Probabilities

"*Dice represent an intriguing example of the interplay between randomness, chance, and physical law.*" (Ivars Peterson's MathTrek [3]).

Ideally, a die should give a fair chance to any of its six sides: the probability of any side landing on a particular face should equal to 1/6. This is for a single die, or a single roll, though. For a double roll, the total of both rolls is distributed in a triangular curve, for three rolls and more, the distribution looks more like the bell-shaped normal distribution (as dictated by the central limit theorem). Figure 2 shows the distributions for an increasing number of dice being rolled.

The objective of this paper is to build a filter on top of our physically imperfect, and therefore unfair, dice. The filter is built in a microcontroller inside the cube, and will monitor the result of the dice rolls, and adjust them if unfair results are noticed. Or in other words: if the behaviour of the die deviates from the expected behaviour of a fair die, its results will be adjusted. The expected behaviour of rolling dice is illustrated in a nutshell in Figure 2; for 1 die it is simply a uniform distribution, for multiple dice, the combined outcome should look more and more like a normal distribution.

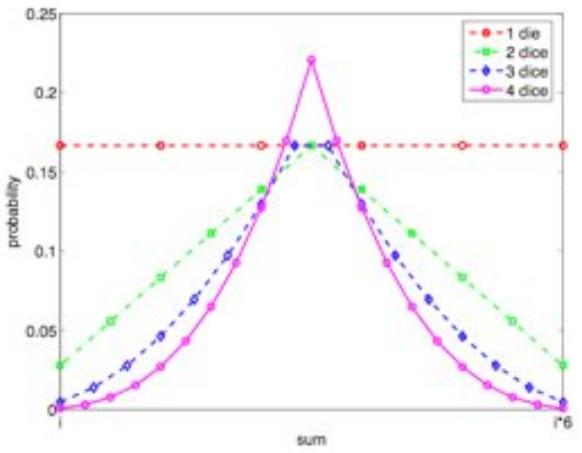

Figure 2. The distributions for the sums of *i* rolled dice, with *i* from 1 to 4. Rolling multiple fair dice result in more likely 'middle' values: the chance of rolling 14 with 4 dice is 0.2207 for instance.

Cristie et al [1] introduced a few parameters to evaluate fairness in dice. *Bias* is defined as:

$$b(j) = n\ \mathrm{p}(j) - 1,$$

for which fair dice have zero bias on all sides. For instance, if a six sided die was so heavily loaded that the six came up half the time (instead of 1/6 of the time) we would say that b(6) = 2.

For dice which have six rectangular faces, the lengths of the sides (defined as d1:6, d2:5 and d3:4) should be equal. The flatness of one of these sides, for example the d1:6 side, is defined as:

$$\mathrm{f1{:}6} = {}^{(\mathrm{d2{:}5} + \mathrm{d3{:}4} - 2\ \mathrm{d1{:}6})}/_{(\mathrm{d2{:}5} + \mathrm{d3{:}4})}$$

If a 20x20x20mm die has 2 mm sawed off the "1" face, then the resulting die has a flatness of f1:6 = 0.1.

Loading is defined as adding mass to one face so as to move the centre of mass away from the geometric centre. An example is drilling a hole in the "2" face, which brings the centre of mass closer to the "5" face. If CM(*j*) is the true location of centre of mass towards face *j*, and *d* the length of all sides, then the loading of side *j* is:

$$\mathrm{L}(j) = 2\ \mathrm{CM}(j) / d$$

L(*j*) is defined so that L(*j*)=0 for fair dice and L(*j*)=1 for the limiting case where the centre of mass is shifted all the way to the *j* face.

5. Implementation

We implemented the fair die on a lightweight wireless sensor node, embedded in a transparent plastic casing (see Figure 1) and molded in epoxy (Figure X). The main module of the hardware consists of a *BSN node* [5], an embedded sensor platform that was developed at Imperial College London for research into body sensor networks, and programmed with Berkeley's TinyOS. A motion- and tilt-specific sensor module, called the *Porcupine*, delivers the sensing, while a minimised battery board holds a 3 Volts coin cell for power. Figure 3 shows a close-up of all three board, stack-connected to each other.

The BSN nodes' main responsibility is the communication of results and the operation of the dice as nodes in a network. For the proposed application, this means the broadcasting of the roll's outcome, and the interface for initial training (discussed later in section 7). The BSN features an 8MIPS Texas Instruments MSP430 ultra low power processor, with a Zigbee ready RF link (Chipcon CC242, 2.4GHz, 250kbps, with hardware MAC encryption, range 50m), and 512Kbytes of EEPROM storage memory.

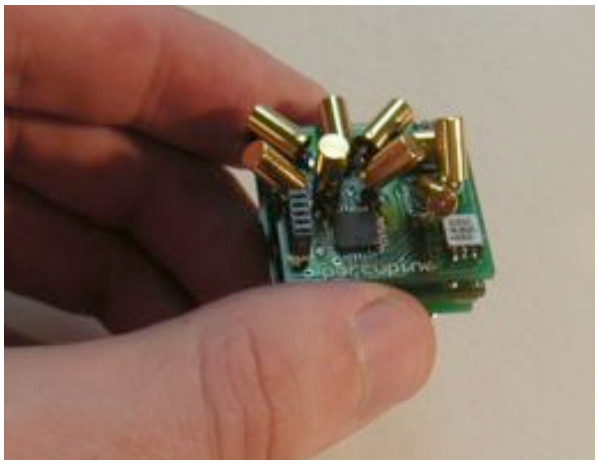

Figure 3. A close-up of the three boards stack-connected to each other. The top ("Porcupine") does the sensing, the middle board (BSN node) the communication, the bottom holds the battery.

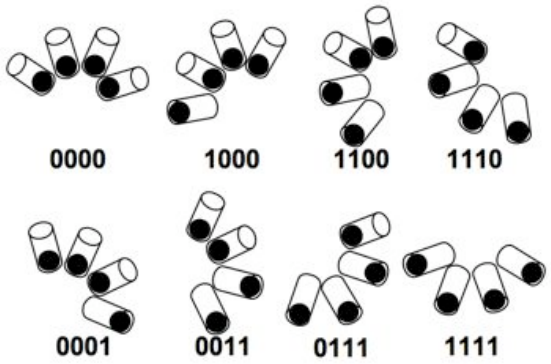

Figure 4. An illustration of the 8 states of tilt that the combination of 4 switches can detect in one plane. Although the tilt switches are bigger and produce less accurate and robust output, they require far less resources than accelerometers.

The Porcupine is a low-power tilt- and motion-sensing board that fuses data from tilt switches and accelerometers, and is able to power down some of its components when they are not necessary. It has a Microchip PIC 16F628A "nanoWatt" microcontroller, 9 ball switches, and 1 Analog Devices ADXL202JE accelerometer module. Although its size is larger than similar sensor boards, its main advantage is that it does its task using a minimal amount of energy.

6. Recognition of rolls and faces

In a previous paper, we used a set of accelerometers in three axes to obtain a cube's tilt and motion relative to gravity [4]. The cube did the processing of sensor data locally, and sent out a message over radio only when its state had changed (for example, if it was turned to another side).

One of the major obstacles with this first prototype was the power consumption: even with the radio sending the messages only occasionally, the rest of the hardware consumed quite a bit of energy. The processing of streaming sensor data went on continuously, meaning that energy was spent even if the cube was not used for long periods. Experiments showed that inducing a sleep mode on the microcontroller at detection of stationary sensor signals would result in a far less responsive system. In a best-case scenario, this system would typically last a few months on 2 AAA batteries, but a lot shorter on smaller coin-cell batteries.

The sensor module of the prototype presented in this paper, called 'Porcupine', is a lot more complex than the one of the previous cube prototype; it contains only one dual-axis accelerometer, but 9 tilt switches that combined give a crude sense of tilt.

6.1. Robust and cheap tilt information

Figure 4 shows how the combined data from 4 switches results in a sense of orientation in one plane. Only 5 additional switches are necessary for the other two perpendicular planes.

The use of the tilt switches has two advantages:

- Energy consumption: They require less power: with the heavy pull-up resistors, they draw only a few microAmperes, compared to a few hundred microAmperes for the accelerometer.
- Processing: Their output is binary, and thus easier and faster to process, especially when combining multiple switches. The microcontroller also doesn't need to run at a fast speed for reading the switches' states or doing analysis.

The microcontroller is able to run at a low speed of 48KHz, and still sampling the tilt switches around 400Hz (including processing). Power consumption of the whole module is in this mode is a low 140 microAmperes.

The mechanical nature of tilt switches introduces several problems, however, that result in occasional errors. Since classifiers on the tilt switch data rely on hamming distance, only one failing switch can easily result in misclassification.

From the analysis of several recorded datasets, where the die was placed on each of its 6 faces (see Figure 5), a few conclusions are made:

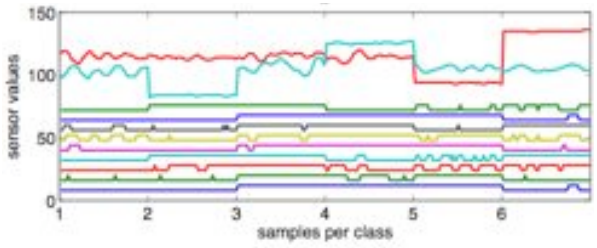

Figure 5. One of the datasets, with sensor samples from the 2 accelerometers (top plots) and the 9 tilt switches (scaled with off-set in the bottom part), per face.

The tilt switches' data for each face recorded only between 4 and 11 sets of unique vectors. This favours lazy classifiers that just store the switches' prototype vectors per face and match new data with these prototypes to estimate the current face (i.e., the current top-side of the die). Only a few duplicates of these prototype vectors were found between faces.

The accelerometer data is very similar for faces 1 and 3; and any recognition algorithm is expected to perform poorly for those. This follows from the fact that only two axes are covered. This is not critical, however, since the switches' data is still very different between faces 1 and 3.

The combination of less reliable tilt switch data and accurate but resource-intensive acceleration data therefore results in a sensor module that produces enough information to distinguish the faces of a die reliably, at an acceptable cost (in processing and energy consumption).

6.2. Estimating faces

By default, the 'Porcupine' sensing module will send the tilt information from the switches, and the last accelerometer information. The latter is not necessarily recent, as the accelerometer is switched off and the module reverts into a low-power mode if there are no fluctuations detected in the sensor signals. As soon as the die is moved (for instance when it is picked up, or when it is rolled), the accelerometers are powered up and accelerometer data are analysed at a higher processing speed.

For estimating which of the 6 faces is pointing upwards, two algorithms are embedded into the die; one for tilt switch data, and one for accelerometer data.

Distance-weighted k nearest neighbours is used for the tilt switches. Experiments have shown that if k is given a high value of 20, recognition rates are reached around 94.9% for all datasets. Using a minimum distance classifier (with the hamming distance, as all vectors have binary components), recognition performance was slightly less (93.5%).

As in a previous version of the cube [4], multivariate Gaussians were used to model accelerometer data per face. The parameters for the model (mean and covariance matrix) are taken from training samples, and estimation of the current face is done by finding the model that is maximised for new inputs.

Figure 6 shows six trained Gaussian models for the six faces; Figure 7 shows the classification performances over a test dataset with accompanying confidence matrices. The results show that overall the performance is very stable for the accelerometer data (multivariate Gaussians), unless the face is 1 or 3. The data from the tilt switches perform better overall, but is less robust as there are a few errors in the classification for almost every face.

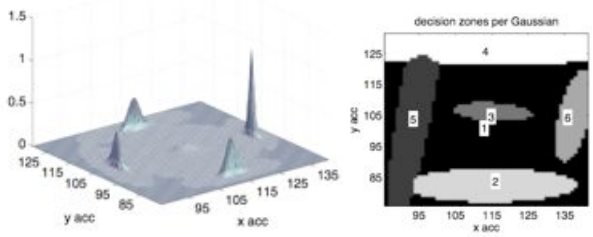

Figure 6. Plots of the multivariate Gaussian models for each of the die's six faces. The left plot shows combined surfaces, the right plot marks the decision areas. Note 1 and 3's overlap.

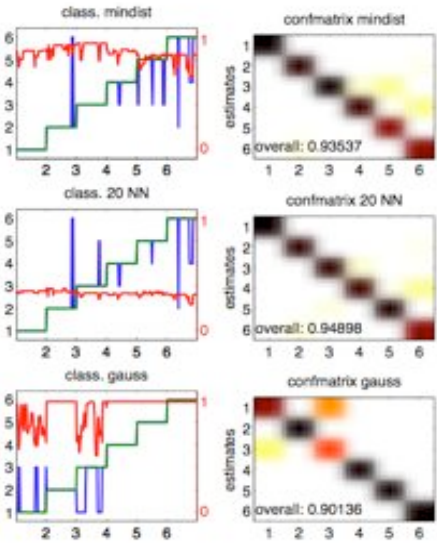

Figure 7. The classification performances over a test dataset with accompanying confidence matrices for (top to bottom): minimum distance, distance-weighted K nearest neighbours, and multivariate Gaussians.

7. Adjusting the behaviour of the dice

One of the key features of our dice is that they are arbitrarily re-trainable, which allows us to embed the hardware in other types of dice (with a different number of faces, for example). Training is done via a wireless node that can communicate with the BSN node inside the dice (typically another BSN node, Figure 1 shows both).

The wireless node can be attached via USB to a phone, PDA or computer as a base station, and this node can be accessed via RS232. The training of a face consists of sending a number over to the base station, which in turn sends it wirelessly to the dice.

When dice classify their sensor data as a different face than previously recognised, a wireless packet is broadcast to any base station in the area, specifying:

- most likely class, estimated by the switch data via a KNN classifier
- confidence value, returned by the switch data via a KNN classifier
- most likely class, estimated by accelerometer data via Gaussian models
- confidence value, returned by accelerometer data via Gaussian models
- the combined most likely class

Additional information can be queried from the dice by sending special commands, such as raw sensor data requests or internal parameters. Any application can thus access this data via the base station. It is important to stress though, that this is a mere interface to the dice, which are themselves responsible for sensing and calculating everything.

The source code (in NESC) for the wireless nodes is the same for the BSN nodes acting as base stations, and the BSN nodes inside the dice. When a Porcupine sensor board is connected to the BSN node, its internal mode switches from base station (acting as an interface between dice and a computer) to die (reading Porcupine data, classifying the data, and logging the die's behaviour). It is also possible to (re-)program the BSN nodes wirelessly inside the dice.

8. Physical properties of dice

The first version of the augmented die was a transparent plastic case, encapsulating the hardware in a near-perfect cube. It became quickly apparent though, that this configuration is not ideal to use in regular board games. This case is prone to cracks when tossed, and – being hollow apart from the wireless sensing node inside – it rolls very irregularly and has the tendency to slide on the surface.

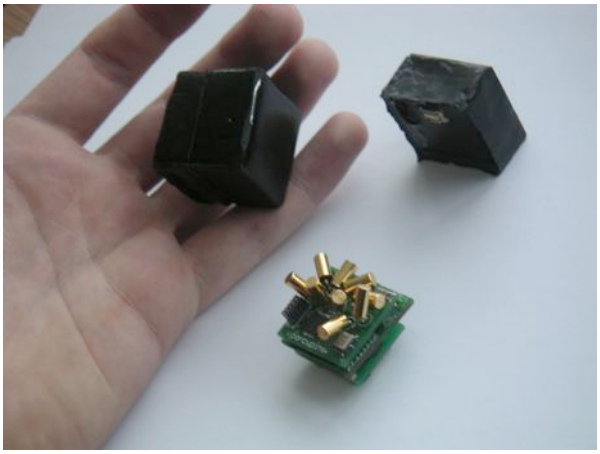

Figure 8. The hardware encapsulated in a epoxide polymer, which makes the cube's weight more distributed, heavier, more homogeneous, and more similar to the dice that people are familiar with in board games.

The current version uses the same hardware, but one that was submerged in a mix of epoxy resin and hardener, and molded in a cubic shape. Figure 8 shows the original hardware boards, and the two halves that make up a heavier, and more robust cubic die, in which the hardware is permanently fixed. The weight and durability of this cube facilitates using it as a die in usability tests.

This epoxy polymer version is molded in two halves that connect to each other (as seen in Figure 8, top-right: one of the two halves): one has the Porcupine sensor board embedded into it, the other contains the battery and BSN node. The microcontrollers on both boards are still re-programmable, and the battery (a 3V CR2032 coin cell battery of 220 milliAmperes) can be replaced through a slit in one of the sides.

9. Initial experiment

As a first short evaluation of our design, and as a test of the reliability of the cube as an augmented die, it was tossed 120 times, as a die in a board game would be tossed. This included trying to ignore the fact that delicate electronics are embedded inside, and trying to let the die roll as much as possible.

Figure 9 shows the die that was rolled in the experiment, with its sides labelled to enable cross-comparison of the output according to the die, and the actual outcome of the roll. After each roll, the evaluator marked the face which the die landed on, which was concatenated after the classification output from the die itself, forming one row in a recorded log that was kept for the duration of the experiment.

Figure 9. Labels were attached to the cube in the experiment to allow cross-checking between the real outcome ("face 2" in this case) and the one classified by the die.

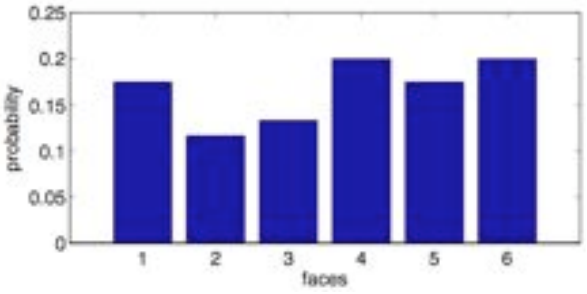

Figure 10. Logged probabilities over 120 trials of tossing the die. The outcome of all 120 tosses was correctly recognized by the sensor node embedded in the die.

The log of all outcomes is also kept in the die itself, so it can calculate its own bias, too. The labels were put on the die in a "clockwise die" arrangement, meaning the faces 1, 2, and 3 are organised from one side in a clockwise manner. The sum of opposite faces is, like in most dice, always seven.

The results (times a face was rolled for 1, 2, 3, 4, 5, and 6) are stored in the 6-dimensional vector:

[21 14 16 24 21 24],

which gives the following probabilities per face (1 to 6):

P(1:6) = [0.1750 0.1167 0.1333 0.2000 0.1750 0.2000]

Although the experiment is statistically very small in size (to evaluate whether the die is biased, many more rolls are required), it is promising that the outcomes estimated by the die were 100% correctly recognised.

10. Conclusions and future work

This paper is concerned with the design of an autonomous cube that detects when it is tossed, and that is able to sense the side on which it has landed, without requiring external components. This 'smart' die is completely re-trainable over wireless communications, and was kept as compact ($30mm^3$) and robust as possible. It is able to retain a history of its past rolls, thus allowing it to find its own bias (i.e., how fair it is).

Further research has already commenced, and will include evaluation and improvement on the recognition of 'fair' rolls (i.e., a die toss which lets the die roll several times), the combining of several dice, and the altering by the die of its own behaviour after finding out its own bias. More extensive evaluations are planned to use this type of dice in real-world situations such as indoor board games.

11. Acknowledgements

The research in this paper was partially funded by CommonSense (EPSRC, UK), and UbiMon (DTI, UK). We would like to thank especially our UbiMon partners at Imperial College, London, for making their BSN platform available for this research.

12. References

[1] Christie, D., Glasheen, R., Hamilton, C., Imoto, M., Matthews, P., Moffat, J., Monajemi, T., Murray, D. B., Nelson, J., and Sturm, A., "Experimentally obtained statistics of dice rolls", poster at the 6th Experimental Chaos Conference, July 22-26, 2001, Potsdam, Germany.

[2] Pegg, E. Jr., "Math Games: Fair Dice", The Mathematical Association of America, May 16, 2005. http://www.maa.org/editorial/mathgames/mathgames_05_16_05.html

[3] Peterson, I., "MathTrek: Unfair Dice" The Mathematical Association of America, October 26, 1998. http://www.maa.org/mathland/mathtrek_10_26_98.html

[4] Van Laerhoven, K., Villar, N., Schmidt, A., Kortuem, G., and Gellersen, H.-W.. "Using an Autonomous Cube for Basic Navigation and Input". In Proceedings of ICMI/PUI 2003. ISBN: 1-58113-621-8; ACM Press. Vancouver, Canada. 2003, pp. 203-211.

[5] Lo, B. and Yang, G. Z., "Key Technical Challenges and Current Implementations of Body Sensor Networks", IEE Proceedings of the 2nd International Workshop on Body Sensor Networks (BSN 2005), pp. 1-5, April 2005.